A Project Report on

Harvest Helper: Smart Agriculture Powered by Blockchain and Machine Learning

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Engineering

in

Information Technology

by

Suyash Utekar (21104066)
Atharva Sutar (21104098)
Yash Shinde (21104029)
Shrikant Pawar (21104086)

Under the Guidance of

Ms. Shraddha Birje

Department of Information Technology
NBA Accredited
A.P. Shah Institute of Technology
G.B.Road,Kasarvadavli, Thane(W)-400615
UNIVERSITY OF MUMBAI

Academic Year 2024-2025

Approval Sheet

This Project Synopsis Report entitled ***"Harvest Helper: Smart Agriculture Powered by Blockchain and Machine Learning"*** Submitted by ***"Suyash Utekar"(21104066), "Atharva Sutar"(21104098), "Yash Shinde"(21104029), "Shrikant Pawar"(21104086)*** is approved for the partial fulfillment of the requirement for the award of the degree of ***Bachelor of Engineering*** in ***Information Technology*** from ***University of Mumbai***.

Ms. Shraddha Birje
Guide

Dr. Kiran Deshpande
HOD, Information Technology

Place:A.P.Shah Institute of Technology, Thane
Date:

CERTIFICATE

This is to certify that the project entitled ***"Harvest Helper: Smart Agriculture Powered by Blockchain and Machine Learning"*** submitted by ***"Suyash Utekar"(21104066), "Atharva Sutar"(21104098), "Yash Shinde"(21104029), "Shrikant Pawar"(21104086)*** for the partial fulfillment of the requirement for award of a degree ***Bachelor of Engineering*** in ***Information Technology***,to the University of Mumbai,is a bonafide work carried out during academic year 2024-2025.

Ms. Shraddha Birje
Guide

Dr. Kiran Deshpande
HOD, Information Technology

Dr. Uttam D.Kolekar
Principal

External Examiner(s)

1.

2.

Internal Examiner(s)

1.

2.

Place:A.P.Shah Institute of Technology, Thane
Date:

Acknowledgement

We have great pleasure in presenting the synopsis report on **Harvest Helper: Smart Agriculture Powered by Blockchain and Machine Learning**. We take this opportunity to express our sincere thanks towards our guide **Ms. Shraddha Birje** for providing the technical guidelines and suggestions regarding line of work. We would like to express our gratitude towards her constant encouragement, support and guidance through the development of project.

We thank **Dr. Kiran B. Deshpande**, Head of Department for his encouragement during the progress meeting and for providing guidelines to write this report.

We express our gratitude towards BE project co-ordinator **Mr. Vishal Badgujar** & **Mr. Sachin Kasare**, for being encouraging throughout the course and for their guidance.

We also thank the entire staff of APSIT for their invaluable help rendered during the course of this work. We wish to express our deep gratitude towards all our colleagues of APSIT for their encouragement.

Suyash Utekar
(21104066)

Atharva Sutar
(21104098)

Yash Shinde
(21104029)

Shrikant Pawar
(21104086)

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)
Suyash Utekar (21104066)

(Signature)
Atharva Sutar (21104098)

(Signature)
Yash Shinde (21104029)

(Signature)
Shrikant Pawar (21104086)

Date:

Abstract

Harvest Helpers is a blockchain-based system designed to enhance transparency, security, and efficiency in agricultural crop supply chain management. The system streamlines the entire process from initial crop application by farmers to the final sale to dealers, involving multiple stakeholders such as farmers, review teams, administrators, and dealers. Through the use of smart contracts, Harvest Helpers automates critical stages including crop verification, mid-term growth assessments, harvest scheduling, grading, and pricing, with all actions securely recorded on the blockchain. Farmers are provided real-time access to track their crop status, book harvest appointments, and approve terms and conditions, while dealers can seamlessly purchase crops through an integrated eCommerce platform. In addition to supply chain management, Harvest Helpers offers a crop and fertilizer prediction feature, leveraging machine learning to suggest optimal crops and fertilizers based on environmental conditions and soil nutrient levels. By utilizing blockchain's decentralized and tamper-proof architecture, Harvest Helpers reduces the possibility of fraud, disputes, and discrepancies, fostering trust and accountability across the agricultural supply chain. This innovative solution not only digitizes crop management and improves traceability but also empowers farmers with greater control and insight into their production, benefitting both small-scale farmers and large agribusinesses alike.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Agriculture is a critical sector that forms the backbone of economies worldwide, particularly in developing countries. However, the agricultural supply chain often faces inefficiencies, lack of transparency, and issues related to trust between stakeholders. Traditional crop management and sale processes are prone to delays, fraud, and disputes, which affect farmers, buyers, and intermediaries alike. These issues lead to disputes over crop quality, pricing, and delivery, ultimately harming farmers' livelihoods and reducing confidence in the supply chain. Furthermore, farmers, particularly smallholder farmers, often lack access to accurate data for crop and fertilizer prediction, making it difficult for them to optimize their yield and profitability. Addressing these problems is crucial for enhancing agricultural productivity and ensuring fairness and transparency across the supply chain.

Harvest Helpers is a blockchain-based platform designed to address these key challenges. The primary aim of Harvest Helpers is to create a transparent, secure, and efficient agricultural supply chain by leveraging blockchain technology and smart contracts. The platform automates critical stages of the crop supply process, including crop verification, mid-term growth assessments, harvest scheduling, grading, and pricing, ensuring that all actions are immutably recorded on the blockchain. By doing so, Harvest Helpers fosters trust among all stakeholders, reducing the likelihood of fraud and disputes while improving the overall efficiency of crop management and sales. Additionally, the system incorporates machine learning algorithms to provide farmers with predictive analytics for crop and fertilizer recommendations based on environmental data and soil conditions, further empowering them to make informed decisions.

The scope of this work extends beyond simply digitizing the supply chain; it aims to offer a holistic solution for agricultural management that benefits all parties involved, from small farmers to large-scale agribusinesses. The significant contributions of Harvest Helpers include its ability to streamline the crop lifecycle, increase accountability through blockchain-based tracking, and enhance decision-making for farmers via machine learning predictions. The system represents an innovative approach to solving longstanding issues in agriculture by combining cutting-edge technologies to improve traceability, efficiency, and fairness in the crop supply chain.

1.1 Motivation

The motivation behind the Harvest Helpers project is rooted in addressing the systemic challenges faced by the agricultural sector, particularly in developing regions where traditional farming practices dominate. These challenges not only affect farmers' livelihoods but also hinder overall agricultural productivity.

- **Inefficiencies in the Supply Chain:**The agricultural supply chain is often plagued by inefficiencies, including delays in processing, lack of transparency, and poor communication among stakeholders. These issues can lead to disputes over pricing and quality, ultimately harming farmers and eroding trust between parties involved in the supply chain.

- **Inefficiencies and Lack of Transparency in Agriculture:**The agricultural sector faces major transparency issues that make existing inefficiencies even worse. Farmers often find it difficult to access important market information, such as prices, demand trends, and consumer preferences. This lack of information makes it hard for them to decide what crops to grow and when to sell them. Additionally, weak traceability systems make it challenging to verify the quality and safety of crops, which can lead to health risks and loss of trust among consumers.

  Slow payment processes and poor quality assurance also create financial problems for farmers, making them more vulnerable to changes in the market. The overall lack of transparency in the supply chain leads to mistrust among all participants, preventing the agricultural sector from developing in a sustainable way.

- **Empowering Farmers with Predictive Analytics:**In addition to enhancing transparency, Harvest Helpers seeks to empower farmers through advanced predictive analytics for crop and fertilizer recommendations. By employing machine learning algorithms to analyze environmental conditions and soil nutrient levels, the platform enables farmers to make informed decisions that can enhance their yields and overall profitability. This data-driven approach aims to bridge the knowledge gap that many farmers face, allowing them to optimize their production practices effectively.

Harvest Helper addresses these inefficiencies and transparency issues in agriculture by implementing blockchain technology to streamline the entire supply chain. The platform enables real-time tracking of crop applications, mid-term checks, and harvest appointments, ensuring that every stakeholder has access to accurate and verified information. Through smart contracts, the system automates critical processes like signing contracts between farmers and the company, grading crops, and recording transactions, reducing delays and manual errors. Harvest Helper empowers farmers with predictive analytics for crop and fertilizer recommendations, helping them make informed decisions based on data-driven insights. Additionally, by creating a transparent platform, Harvest Helper fosters trust between farmers, review teams, and dealers, enhancing collaboration and reducing the risk of fraud. The integrated eCommerce feature connects farmers directly with buyers, ensuring fair pricing and secure payment upon successful delivery of crops.

1.2 Problem Statement

The agricultural sector faces significant challenges characterized by inefficiencies, lack of transparency, and inadequate access to market information, leading to financial instability for farmers and mistrust among stakeholders. Farmers struggle to obtain critical market insights, resulting in poor decision-making regarding crop selection and timing for sales. Additionally, the fragmented supply chain, compounded by multiple intermediaries, leads to increased costs and delays in product delivery. The absence of reliable quality assurance measures and traceability mechanisms further exacerbates these issues, posing risks to food safety and consumer trust. Moreover, traditional practices hinder farmers' ability to leverage modern technologies, such as predictive analytics, that could enhance productivity and sustainability. To address these challenges, there is a need for an integrated solution that utilizes blockchain technology to improve transparency, streamline processes, and empower farmers with data-driven insights, thereby fostering a more resilient and efficient agricultural ecosystem.

1.3 Objectives

- To streamline supply chain processes, automate critical supply chain tasks through smart contracts to reduce delays, manual errors, and administrative burdens in the agricultural supply chain.

- To facilitate direct market access, create an integrated eCommerce platform that connects farmers directly with buyers, ensuring fair pricing and timely payments upon successful delivery of crops.

- To enhance transparency, implement a blockchain-based system that provides tracking of crop applications, mid-term checks, and harvest data to ensure all stakeholders have access to accurate and verifiable information.

- To empower farmers with data-driven insights, provide them with predictive analytics for crop and fertilizer predictions, enabling informed decisions that optimize productivity and resource management.

- To improve quality assurance and traceability, establish robust quality assurance measures and traceability mechanisms to enhance food safety, boost consumer confidence, and prevent fraud within the agricultural supply chain.

- To foster collaboration among stakeholders, promote better communication and collaboration among farmers, review teams, dealers, and administrators to build trust and strengthen the agricultural ecosystem.

- To create an easy-to-use interface, design user-friendly interfaces for all stakeholders—including farmers, dealers, administrators, and review teams—to ensure ease of navigation and efficient interaction with the platform, fostering widespread adoption and satisfaction.

1.4 Scope

- Can involve multiple stakeholders in the agricultural supply chain, including farmers, dealers, administrators, and review teams, ensuring that their needs and perspectives are integrated into the system design.
- Can implement blockchain technology to enhance transparency, security, and traceability within the supply chain, allowing for immutable records of transactions and interactions.

- Can incorporate predictive analytics to offer insights on crop yields, fertilizer requirements, and market trends, enabling stakeholders to make data-driven decisions.

- Can develop an integrated eCommerce module to facilitate direct transactions between farmers and dealers, streamlining the sales process and improving market access for farmers.

- Can establish processes for quality control and assurance to allow effective verification of crop quality and safety throughout the supply chain.

- Can focus on creating intuitive and user-friendly interfaces for all stakeholders, ensuring that users can easily navigate the system and access relevant features.

Chapter 2

Literature Review

Sr No	Paper Title	Authors	Key Findings
1	Transforming agricultural supply chains: Leveraging blockchain-enabled java smart contracts and IoT integration (2024)	**E. N. Chuka, B. A. Alhassan, and P. C. Ndama**	Blockchain ensures transparency and security in agricultural supply chains by creating immutable records of transactions and tracking products from farm to consumer. Smart contracts automate decision-making processes, such as verifying conditions for product quality, payment, and delivery, reducing the need for manual oversight. By eliminating intermediaries and providing tamper-proof data, blockchain fosters trust between stakeholders and enhances overall supply chain efficiency.

Table 2.1: Literature Review - 1

Sr No	Paper Title	Authors	Key Findings
2	Blockchain Enabled Quality Management in Short Food Supply Chains (2022)	**Patrick Burgess, Funlade Sunmola, and Sigrid Wertheim-Heck**	Blockchain technology can significantly enhance quality management in short food supply chains by providing transparent and traceable records of product handling and processing. It facilitates the establishment of quality standards and governance structures that are tailored to local contexts, ensuring that food quality meets consumer expectations. By integrating IoT devices, stakeholders can collect real-time data on product conditions, allowing for timely interventions and improved decision-making. Additionally, blockchain fosters trust among stakeholders by ensuring the integrity of data, reducing the risk of fraud, and promoting ethical practices throughout the supply chain. Overall, these features contribute to improved sustainability, accountability, and consumer confidence in locally sourced food products.

Table 2.2: Literature Review - 2

Sr No	Paper Title	Authors	Key Findings
3	Blockchain-Based Agricultural Supply Chain: A Review (2021)	**Zhang Y., P. White, D. Schmidt, and L. Louw**	"Blockchain-Based Agricultural Supply Chain: A Review" indicate that blockchain technology enhances transparency, traceability, and efficiency in agricultural supply chains. It provides immutable records of transactions to ensure product authenticity and safety, reducing food fraud risks. Smart contracts automate the enforcement of agreements and quality standards, streamlining processes like payment and delivery. The integration of IoT devices allows for real-time monitoring of product conditions, improving resource management. Overall, these advancements foster increased stakeholder trust, sustainability, and operational efficiency in agricultural supply chains.

Table 2.3: Literature Review - 3

Chapter 3

Project Design

3.1 Existing System

The current agricultural supply chain faces multiple inefficiencies due to its reliance on traditional, centralized systems. Below are key aspects of the existing system:

Manual Crop Application and Verification

- Farmers typically apply for crop verification through government agencies or intermediaries.
- These applications are processed manually, leading to delays and inefficiencies.
- Verification is prone to errors and fraud due to a lack of transparency.

Lack of Real-Time Crop Monitoring

- Farmers and regulatory bodies do not have a real-time system for tracking crop growth.
- Traditional methods rely on field inspections, which are time-consuming and inconsistent.
- There is no standardized platform for scheduling mid-term assessments.

Opaque Pricing Mechanism

- Crop pricing is often determined by intermediaries rather than market demand.
- There is no transparent mechanism to verify grading and pricing decisions.

Limited Digitization and Data Management

- Most agricultural transactions are recorded on paper, leading to data loss and inefficiencies.
- There is no centralized system to store crop history, making traceability difficult.

Susceptibility to Fraud and Manipulation

- Due to the absence of an immutable system, records can be manipulated by intermediaries.
- Farmers may receive delayed payments, and disputes over crop quality are common.
- Buyers have limited means to verify whether crops meet promised quality standards.

No Direct Market Access for Farmers

- Farmers must rely on middlemen or government-controlled platforms to sell crops.
- Direct transactions between farmers and buyers are rare, reducing profitability for producers.
- Existing systems do not offer an integrated platform for eCommerce-based crop sales.

By addressing these inefficiencies, Harvest Helper aims to introduce blockchain and machine learning to enhance transparency, automate processes, and provide real-time monitoring and pricing mechanisms.

3.2 Proposed System

The proposed system architecture for the Harvest Helpers project follows a multi-layered approach designed to facilitate seamless interaction among stakeholders in the agricultural supply chain. At the User Interface Layer, farmers, dealers, review teams, and administrators will each have tailored, user-friendly interfaces built with Flask, a lightweight web framework that enables dynamic and responsive applications. This allows for smooth submission of crop applications, browsing of available crops, verification of information, and management of user accounts.

The Application Layer incorporates smart contracts that automate actions based on specific triggers, a data analytics module providing predictive insights on crops and fertilizers, and a notification system to alert users of important updates. Data is securely stored in a MongoDB database, which provides a flexible schema to accommodate the diverse types of data generated by the system, including user profiles, crop data, historical transactions, and analytical data. Additionally, a dedicated media storage solution is used to handle images and documents linked to transactions.

At the core of the architecture is the Blockchain Layer, featuring a distributed ledger that ensures all transactions and data inputs are recorded immutably and transparently, along with a consensus mechanism that maintains the integrity and consistency of the blockchain. The Security Layer implements user authentication and authorization mechanisms to ensure secure access, along with encryption protocols that protect sensitive data in transit and at rest.

The overall flow of the system begins with farmers submitting crop applications, which are verified by the review team and recorded on the blockchain. Automated notifications guide mid-term checks and harvest appointments, leading to secure crop sales and payments. This comprehensive architecture promotes a secure, efficient, and transparent agricultural supply chain, benefiting all participants and enhancing overall productivity in the sector.

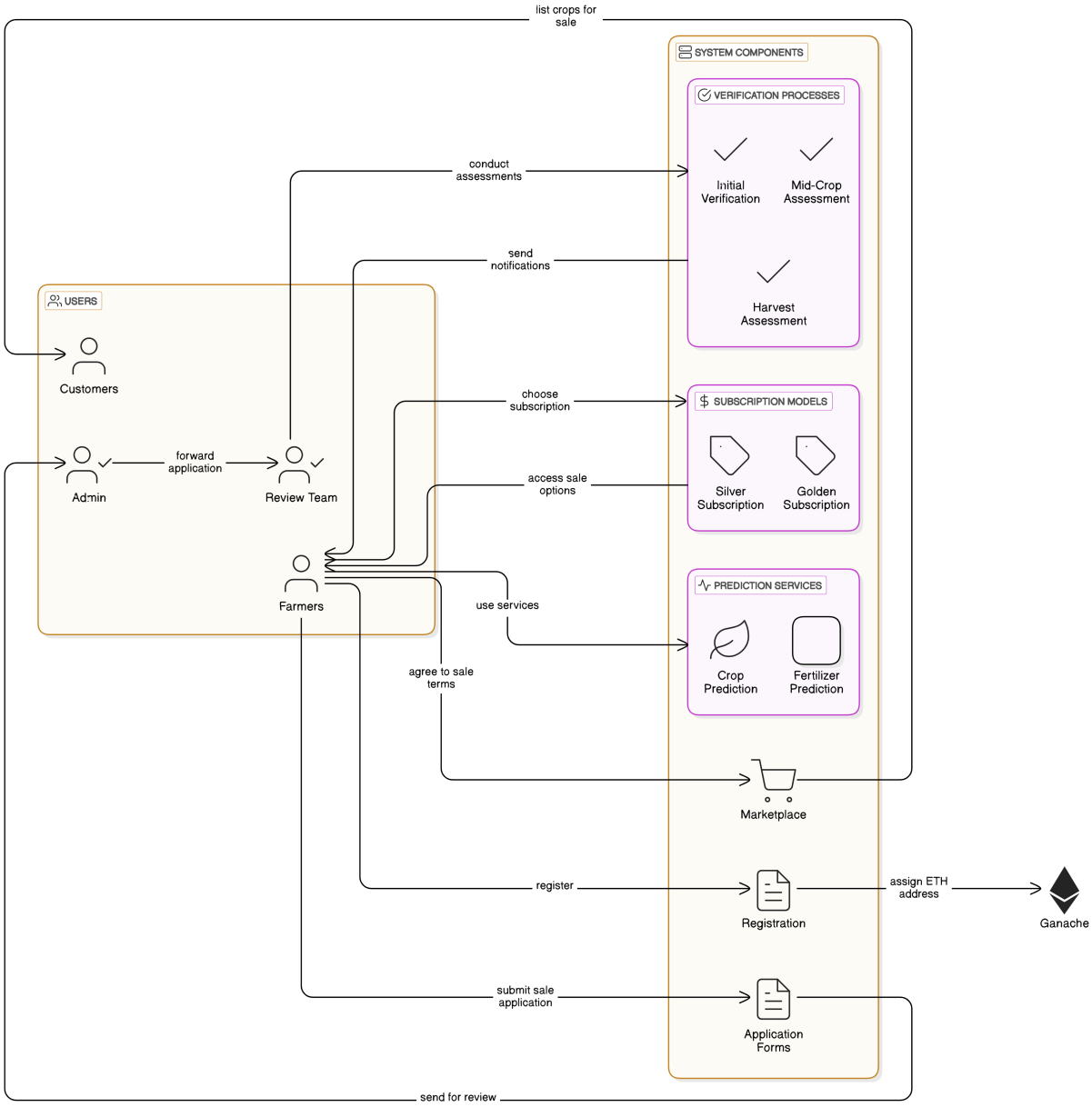


Figure 3.1: System Architecture

3.2.1 Critical Components of System Architecture

The **Harvest Helper Project** is a decentralized application (DApp) that integrates blockchain technology with a crop sale and assessment system. It facilitates secure transactions, transparent verification processes, and efficient crop management. The system architecture consists of multiple components, each playing a crucial role in ensuring seamless functionality and user interactions.

User Management

The system includes four types of users:

- **Farmers**: Farmers register in the system and are assigned a unique Ethereum (ETH) address using Ganache. They can access various services such as crop prediction, fertilizer prediction, and crop sale.
- **Administrators**: The admin team manages users, processes crop sale applications, and oversees system operations.
- **Review Team**: This team is responsible for verifying farmer-submitted details through multiple assessment stages (initial verification, mid-crop assessment, and harvest assessment).
- **Customers**: Customers can browse the marketplace, view crop details, and purchase crops listed by farmers.

Registration and Authentication

- Farmers must register by providing personal details such as name, email, username, and password.
- During registration, each farmer is assigned a unique ETH address using Ganache, ensuring blockchain-based identity verification.
- The authentication system ensures secure login and access control for different user roles.

Prediction Services

The system provides AI-powered prediction services to help farmers make data-driven decisions.

- **Crop Prediction**: Based on soil parameters (nitrogen, phosphorus, potassium, temperature, humidity, pH, and rainfall), the system predicts the most suitable crop for cultivation.
- **Fertilizer Prediction**: Based on soil nutrient levels, the system suggests appropriate fertilizers to maximize crop yield.

Subscription Models

Farmers must subscribe to avail themselves of crop sale services. Two subscription models are available:

- **Silver Subscription**: Costs INR 500 and provides access to basic crop sale features.
- **Golden Subscription**: Costs INR 1000 and includes additional benefits such as insurance coverage.

Verification and Assessment

The crop verification process involves multiple assessments by the review team:

- **Initial Verification**: The review team verifies farmer-submitted details before approving the sale application.
- **Mid-Crop Assessment**: Conducted within 90 days of verification to assess crop growth and assign grades (A, B, or C).
- **Harvest Assessment**: Performed before harvesting to finalize crop quality, determine the sale price, and record crop quantity.

Blockchain Integration

- Each registered farmer is assigned an Ethereum address through Ganache.
- Transactions related to crop sales and verification are stored securely on the blockchain.
- Ensures transparency and immutability of data.

Marketplace and Crop Sale

Once the verification process is complete, farmers can list their crops for sale.

- Farmers agree to the sale terms before listing crops.
- The admin reviews and approves crop listings.
- Customers can browse the marketplace and purchase crops with full visibility into crop history, assessment grades, and harvest details.

Notifications and Messaging

- Farmers receive email notifications at various stages, such as application approval, assessment reminders, and sale confirmation.
- System-generated alerts guide farmers through the verification and assessment process.

3.3 System Diagrams

3.3.1 UML Diagram

Unified Modeling Language (UML) diagrams are essential tools for visually representing the structure and behavior of a system. They provide a standardized notation to depict system components, their relationships, and interactions.In the context of Harvest Helper, UML diagrams play a crucial role in illustrating the system architecture, workflows, and user interactions.Several types of UML diagrams are utilized to capture different aspects of the system. These include class diagrams, use case diagrams, sequence diagrams, and activity diagrams. Each of these diagrams contributes to a comprehensive understanding of the system's design and functionality.

3.3.2 Activity Diagram

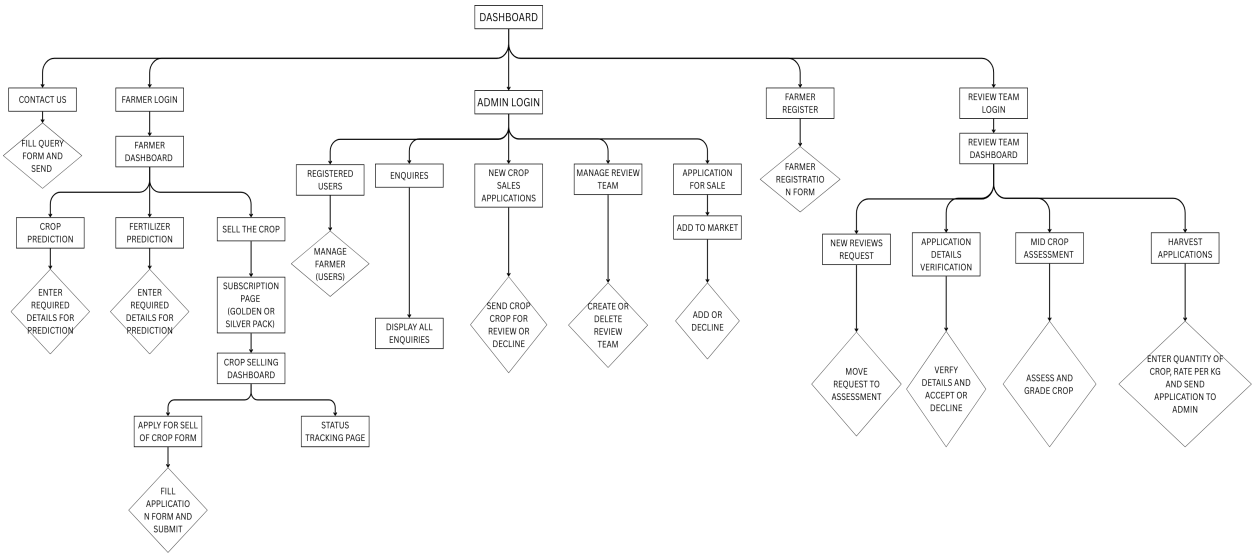


Figure 3.2: Activity Diagram

The activity diagram Figure 3.2 represents the complete workflow of the blockchain-based crop supply chain system, capturing interactions between different user roles — Farmers, Admins, and Review Teams.

In the Farmer Module, farmers can log in to a dashboard that offers services such as crop prediction, fertilizer prediction, and the option to sell crops. To initiate the selling process, a farmer first selects a subscription plan, either Golden or Silver. After that, they gain access to the Crop selling dashboard where they can submit a crop sale application. Farmers can also track the status of their submitted applications through a dedicated status tracking page.

In the Admin Module, administrators manage various aspects of the platform. They oversee the list of registered farmers, view incoming enquiries, handle new crop sale applications,

and assign those applications to appropriate review teams. When a new crop application is received, the admin either forwards it for further review or declines it based on the initial assessment. Applications that are approved are then added to the marketplace, making them publicly visible for potential buyers.

The Review Team Module comes into play once the applications are forwarded for review. Review team members log in to access their dashboard, which allows them to manage new reviews, verify submitted application details, and decide whether to accept or decline them. They also conduct mid-crop assessments where crops are examined and graded. When the crops are ready for harvest, the review team enters the final crop quantity and the per-kilogram rate, then forwards this information back to the admin.

Farmer registration is made easy through a dedicated registration form, allowing new users to sign up. Additionally, there is a contact or query form that enables any user to reach out to the platform with questions or concerns.

This structured activity diagram ensures a clear understanding of the system's operations and promotes smooth coordination among all stakeholders. It enables effective tracking, evaluation, and marketing of crops using blockchain technology.

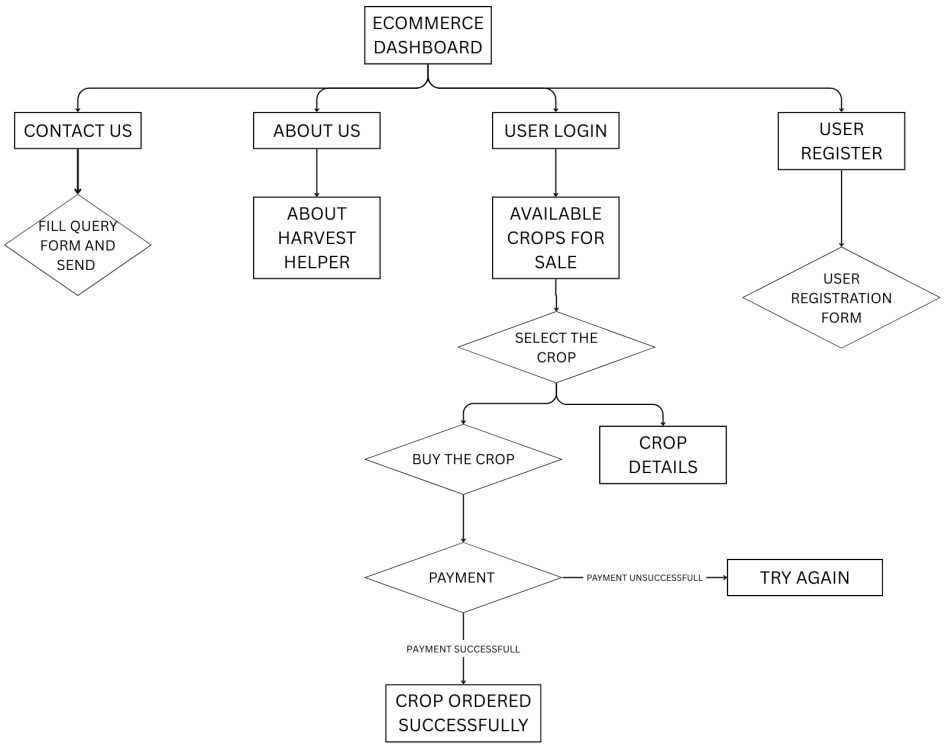


Figure 3.3: Activity Diagram of Ecommerce

This activity diagram Figure 3.3 illustrates the workflow of the E-commerce module of the crop supply chain system, focusing on the process of crop purchasing by users.

Users begin by accessing the E-commerce Dashboard, which offers several options. The "Contact Us" feature allows users to submit queries through a form, while the "About Us" section provides information about the system under the heading "Harvest Helper". Users

also have the ability to log in or register through a dedicated form that facilitates account creation.

Once a user has logged in, they can browse the list of crops available for sale. Upon selecting a crop, they may choose to view more detailed information or proceed directly to purchase.

During the purchase process, the system guides the user to the payment step. If the payment is completed successfully, the system confirms the crop has been ordered. However, if the payment fails, the user receives a prompt encouraging them to try again.

New users must complete a registration form before they are granted access to purchase features. This step ensures that only registered users can engage in crop buying activities.

Overall, this diagram helps visualize a seamless and user-friendly interaction flow for buyers, covering every step from registration to the final crop purchase. It ensures clarity and provides a smooth experience for users navigating the e-commerce section of the platform.

3.3.3 Use Case Diagram

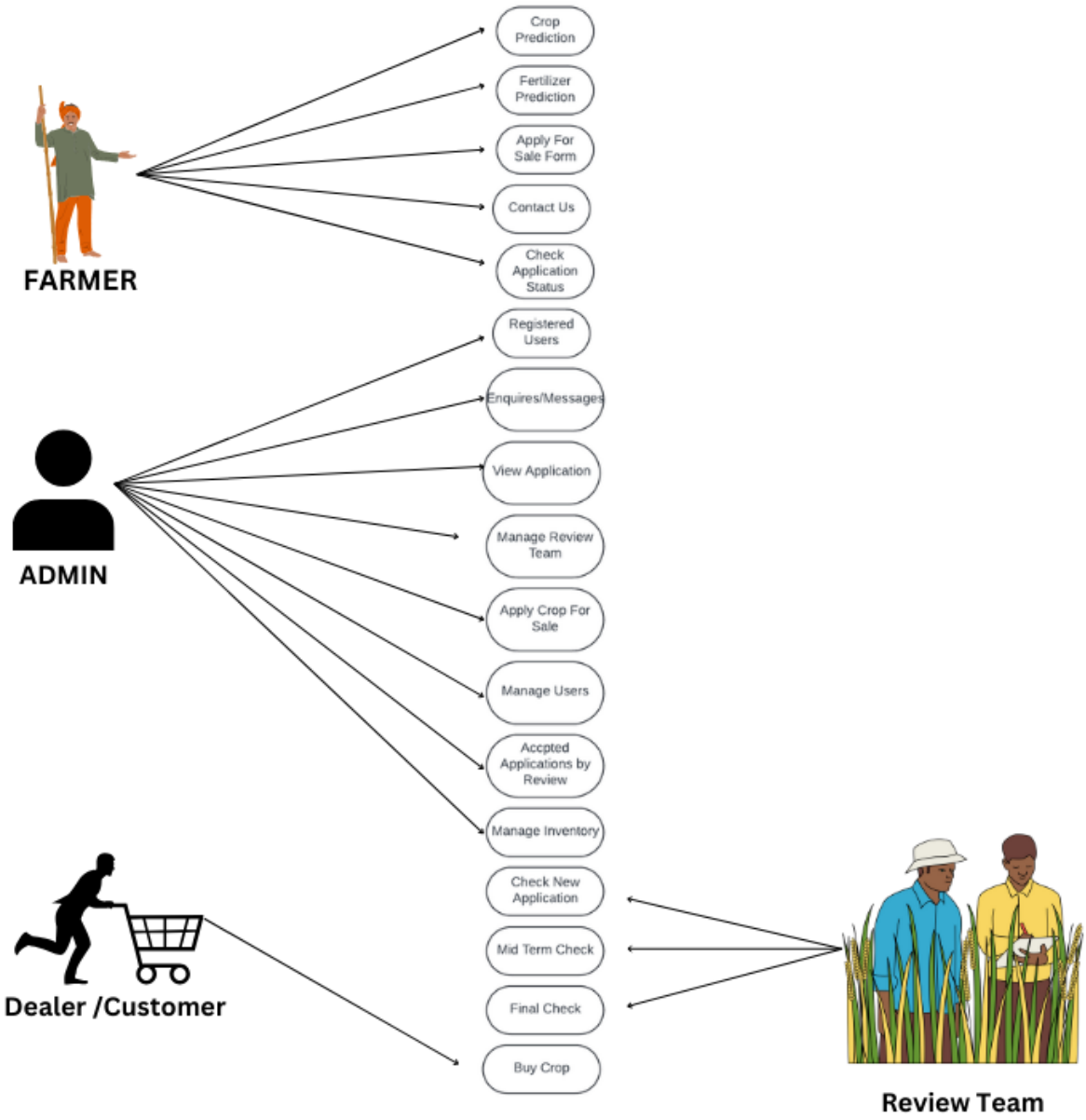


Figure 3.4: Use Case Diagram

Harvest Helpers platform illustrates the interactions between various stakeholders, namely Farmers, Admins, Dealers/Customers, and the Review Team, along with the key functionalities accessible to each. Farmers utilize the platform for tasks such as crop and fertilizer prediction, applying for crop sales, checking the status of their applications, and contacting the admin through inquiry features. Admins play a central role in managing the system, handling user registrations, viewing and reviewing applications, managing the review team, and overseeing inventory. They also decide on the acceptance or rejection of crop sale requests. The review team is responsible for inspecting crops during different

stages, including the initial check, mid-term check, and final inspection, and then reporting their findings to the admin. Dealers and customers interact with the platform primarily by purchasing crops listed on the eCommerce section. This structured interaction among the stakeholders ensures efficient crop management, transparent communication, and smooth transactions within the supply chain.

3.3.4 Data Flow Diagrams (DFD)

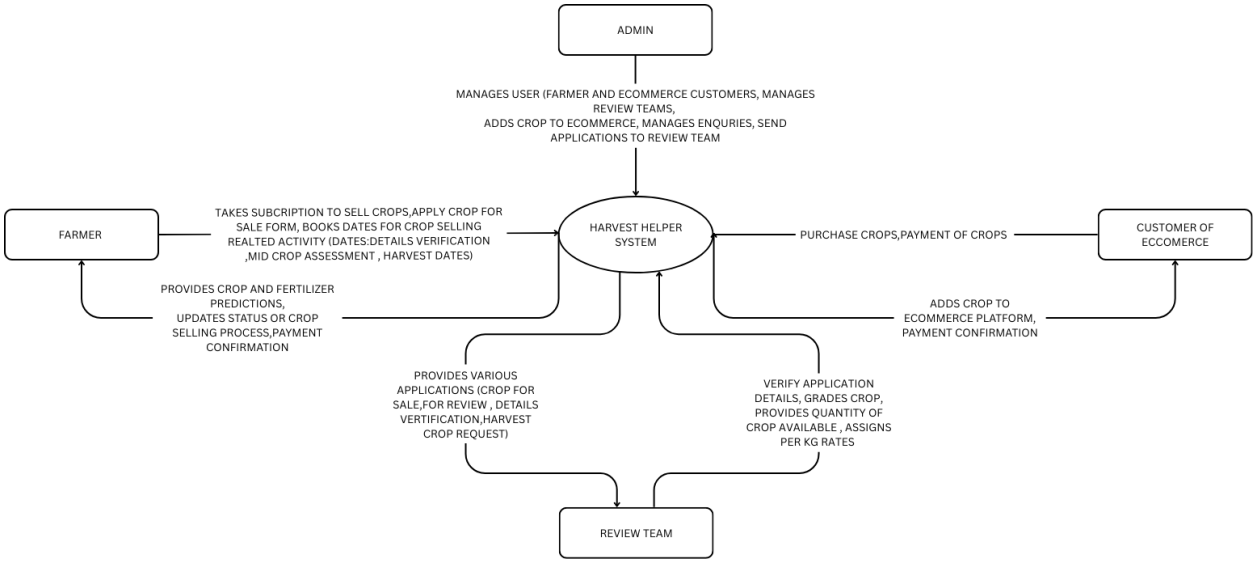


Figure 3.5: DFD LEVEL-0

The Level 0 Data Flow Diagram (DFD) Figure 3.5 for the Harvest Helper System illustrates the high-level interactions between the central system and its external entities. The core process, labeled "Harvest Helper System," is the central component that connects and facilitates all major operations. The Farmer interacts with the system to subscribe for crop selling, apply for sales, and book dates related to crop activities such as detail verification, mid-crop assessment, and harvesting. Farmers also receive crop and fertilizer predictions, status updates, and payment confirmations from the system.

The Admin manages both the farmers and e-commerce customers, oversees the review team, adds crops to the e-commerce platform, manages user inquiries, and forwards applications to the review team for verification. The Review Team plays a crucial role by verifying application details, grading the crops, providing available crop quantities, and assigning per-kg rates. They also submit various applications such as crop-for-sale forms, review requests, verification details, and harvest-related crop data.

On the buyer side, the Customer of E-Commerce interacts with the system by purchasing crops and making payments. The system, in turn, confirms their payments and provides access to listed crops. Overall, the DFD captures how the Harvest Helper System streamlines the crop selling lifecycle, ensures transparency, and bridges communication between farmers, admin, reviewers, and customers, forming a cohesive digital agricultural marketplace.

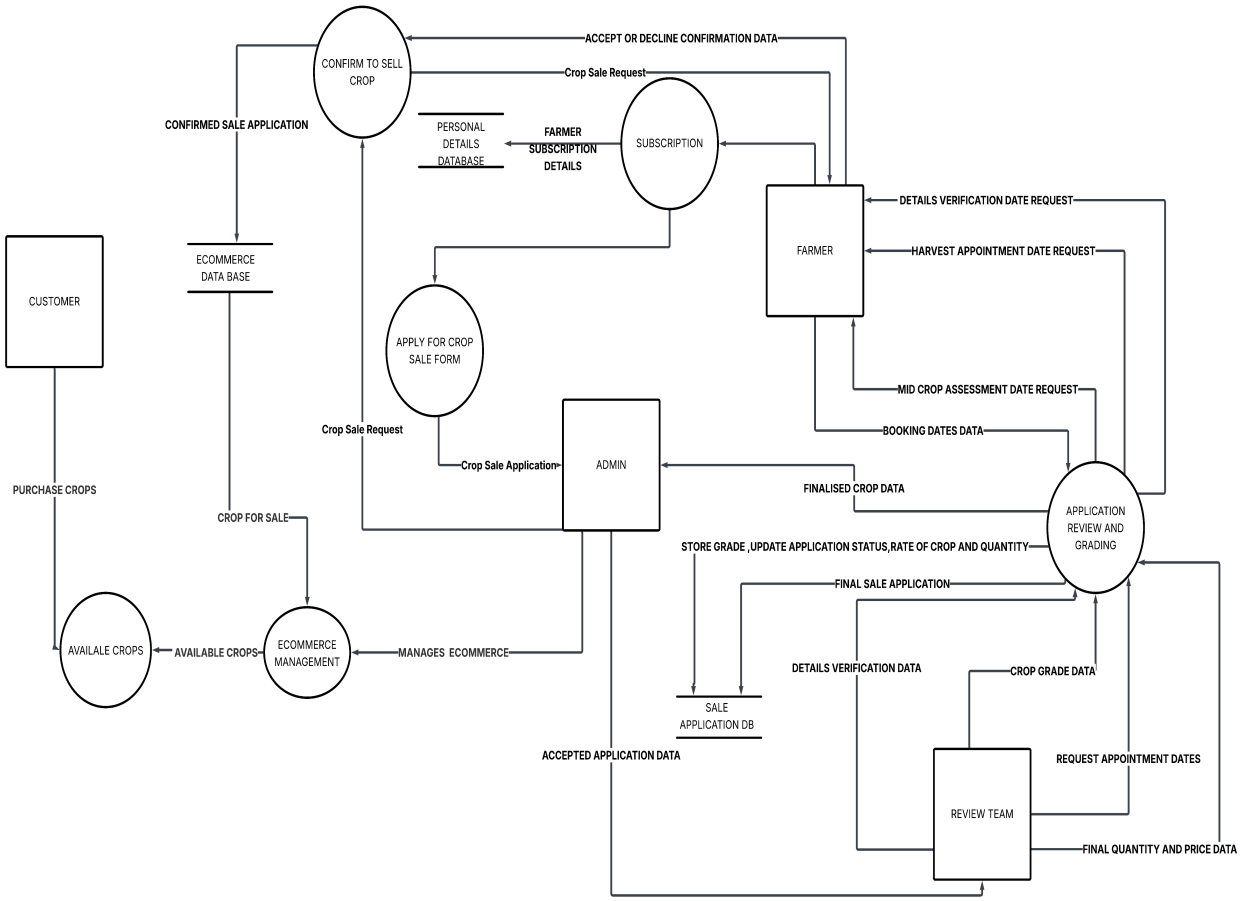


Figure 3.6: DFD LEVEL-1

The Level 1 Data Flow Diagram (DFD) Figure 3.6 for the Harvest Helper Project outlines the essential processes involved in managing the crop sale lifecycle, beginning with farmer registration and ending with customer purchase. The process starts when a farmer submits a crop sale application form, which includes detailed information such as crop name, crop type, location, and ownership. This application is routed to the admin, who forwards it to the review team for evaluation. The review team assesses the application and initiates requests for verification appointments. The farmer then uses the dashboard to book appointments for details verification, mid-crop assessment, and harvest. Following this, the review team conducts field visits and updates the system with verified data, crop grading, and the final quantity and pricing details. This finalized data is reviewed by the admin and stored in the sale application database. Once the application is approved, the farmer confirms the sale, and the crop details are transferred to the e-commerce database.Customers can browse the e-commerce section and buy listed crops.Customers also gain visibility into the crop's history, grading, and assessments. Overall, the system ensures a transparent farm-to-consumer journey, validating each step through booking data and internal checks.

Chapter 4

Project Implementation

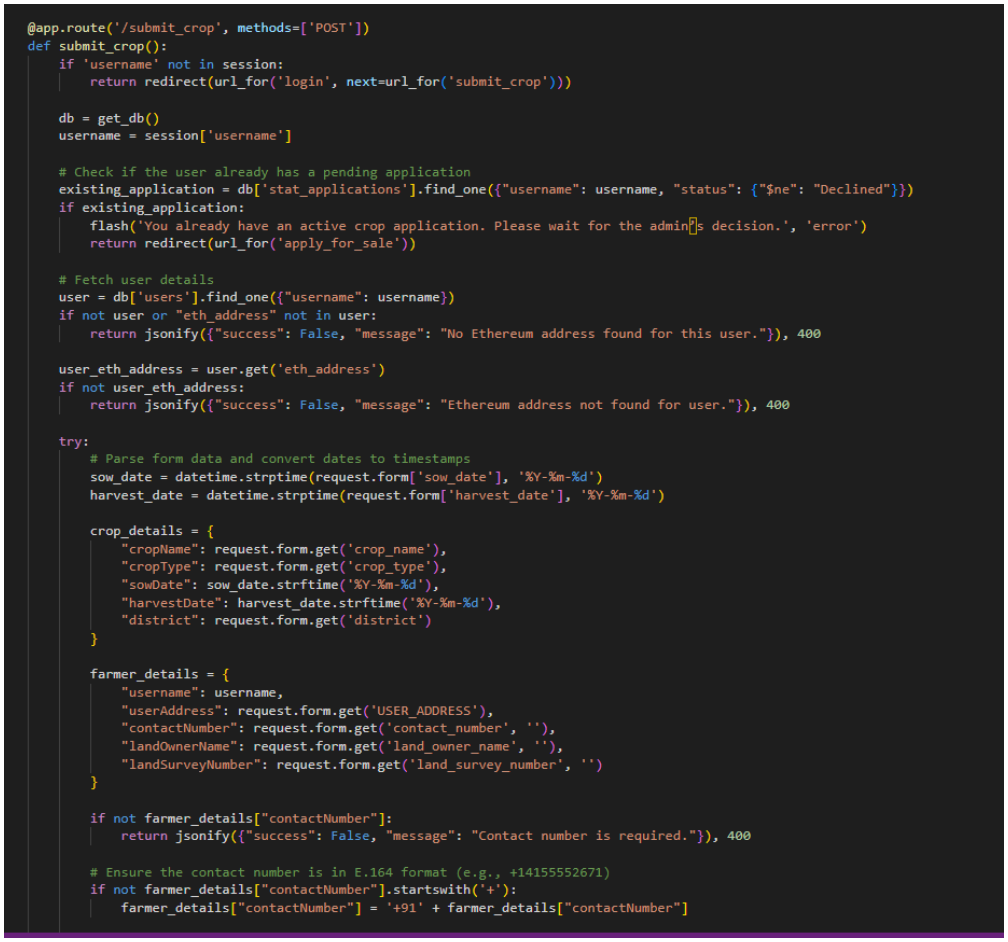
4.1 Code Snippets

C: > test main > Crop recommendation system > crop_app > contracts > CropApplication.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract CropApplication {
    struct Application {
        bytes32 id;
        address ethAddress;
        string status;
        uint midCropAssessmentDate;
        uint harvestDate;
        string cropGrade;
        uint perKgRate;
        bool graded;
        bool rateAssigned;
        uint gradingTimestamp;
        uint rateTimestamp;
        uint quantity;
    }

    mapping(bytes32 => Application) public cropApplications;
    mapping(address => bytes32) public ethToApplicationId;
    bytes32[] public applicationIds;

    event CropApplicationSubmitted(bytes32 id, address indexed ethAddress, string username, string status);
    event ApplicationStatusUpdated(bytes32 id, address indexed ethAddress, string status);
    event MidCropAssessmentBooked(bytes32 id, address indexed ethAddress, uint assessmentDate);
    event HarvestDateBooked(bytes32 id, address indexed ethAddress, uint harvestDate);
    event CropGraded(bytes32 id, address indexed ethAddress, string grade, uint timestamp);
    event PerKgRateAssigned(bytes32 id, address indexed ethAddress, uint rate,uint quantity, uint timestamp);
    event ApplicationMovedToMidAssessment(bytes32 id, address indexed ethAddress, string status);
    event ApplicationMovedToHarvestStage(bytes32 id, address indexed ethAddress, string status);
```

Figure 4.1: Smart Contract Solidity Code

Figure 4.1 is CropApplication.sol smart contract which manages crop applications, mid-crop assessments, harvest bookings, grading, and pricing on the blockchain. Farmers submit applications, which are verified by the review team. After verification, the farmer books a mid-crop assessment date, and on that date, the review team grades the crop. The grade is stored on the blockchain along with a timestamp. Next, the farmer books a harvest date, and on that date, the team assigns a per KG rate. The rate and quantity details are recorded on-chain for transparency. Smart contract events automatically update the database in Flask. Farmers and buyers can track the full crop history via blockchain. This ensures transparency, automation, and security in the Harvest Helper project.



```python
@app.route('/submit_crop', methods=['POST'])
def submit_crop():
    if 'username' not in session:
        return redirect(url_for('login', next=url_for('submit_crop')))

    db = get_db()
    username = session['username']

    # Check if the user already has a pending application
    existing_application = db['stat_applications'].find_one({"username": username, "status": {"$ne": "Declined"}})
    if existing_application:
        flash('You already have an active crop application. Please wait for the admin's decision.', 'error')
        return redirect(url_for('apply_for_sale'))

    # Fetch user details
    user = db['users'].find_one({"username": username})
    if not user or "eth_address" not in user:
        return jsonify({"success": False, "message": "No Ethereum address found for this user."}), 400

    user_eth_address = user.get('eth_address')
    if not user_eth_address:
        return jsonify({"success": False, "message": "Ethereum address not found for user."}), 400

    try:
        # Parse form data and convert dates to timestamps
        sow_date = datetime.strptime(request.form['sow_date'], '%Y-%m-%d')
        harvest_date = datetime.strptime(request.form['harvest_date'], '%Y-%m-%d')

        crop_details = {
            "cropName": request.form.get('crop_name'),
            "cropType": request.form.get('crop_type'),
            "sowDate": sow_date.strftime('%Y-%m-%d'),
            "harvestDate": harvest_date.strftime('%Y-%m-%d'),
            "district": request.form.get('district')
        }

        farmer_details = {
            "username": username,
            "userAddress": request.form.get('USER_ADDRESS'),
            "contactNumber": request.form.get('contact_number', ''),
            "landOwnerName": request.form.get('land_owner_name', ''),
            "landSurveyNumber": request.form.get('land_survey_number', '')
        }

        if not farmer_details["contactNumber"]:
            return jsonify({"success": False, "message": "Contact number is required."}), 400

        # Ensure the contact number is in E.164 format (e.g., +14155552671)
        if not farmer_details["contactNumber"].startswith('+'):
            farmer_details["contactNumber"] = '+91' + farmer_details["contactNumber"]
```

Figure 4.2: Crop Application Submit

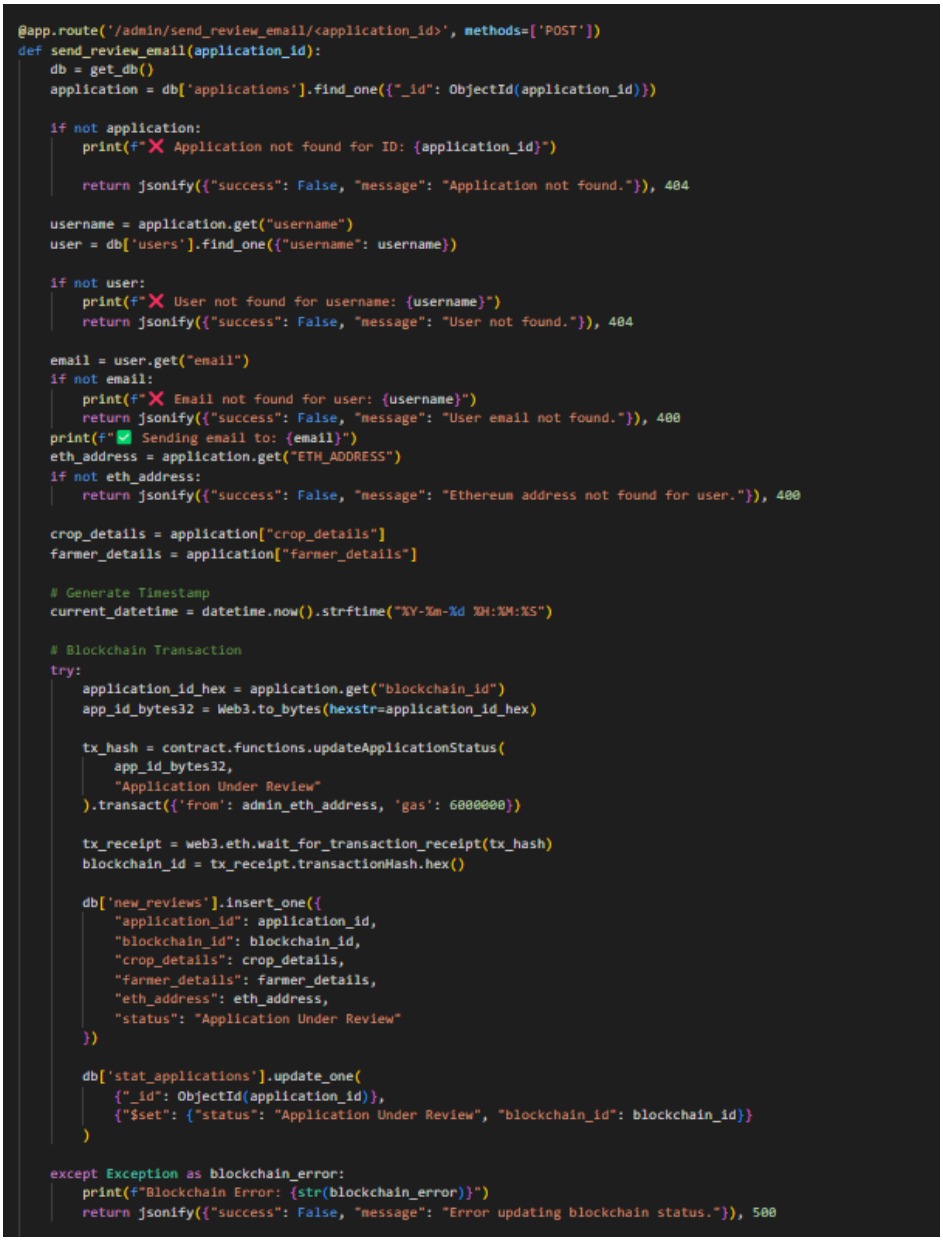
The submitcrop route handles the submission of crop sale applications by farmers. It begins by verifying that the user is logged in via session. It checks MongoDB (statapplications collection) to ensure the farmer doesn't already have a pending application, preventing duplicates.

Then, it retrieves the user's Ethereum address and parses the form data, including sow and harvest dates. These dates are converted to UNIX timestamps since smart contracts use numeric values.

User and crop details are prepared and passed to the submitApplication smart contract method using the Web3 library. The transaction is sent to the Ethereum blockchain using the user's Ethereum address. Once confirmed, the function fetches the application ID (as bytes32) from the smart contract and converts it into a readable hex string.

Finally, all application data, including the blockchain ID and user info, is stored in two MongoDB collections (applications and statapplications). If successful, the user is redirected to their dashboard.

This function ensures seamless interaction between the web interface, MongoDB, and Ethereum smart contract.



```python
@app.route('/admin/send_review_email/<application_id>', methods=['POST'])
def send_review_email(application_id):
    db = get_db()
    application = db['applications'].find_one({"_id": ObjectId(application_id)})

    if not application:
        print(f"❌ Application not found for ID: {application_id}")

        return jsonify({"success": False, "message": "Application not found."}), 404

    username = application.get("username")
    user = db['users'].find_one({"username": username})

    if not user:
        print(f"❌ User not found for username: {username}")
        return jsonify({"success": False, "message": "User not found."}), 404

    email = user.get("email")
    if not email:
        print(f"❌ Email not found for user: {username}")
        return jsonify({"success": False, "message": "User email not found."}), 400
    print(f"✅ Sending email to: {email}")
    eth_address = application.get("ETH_ADDRESS")
    if not eth_address:
        return jsonify({"success": False, "message": "Ethereum address not found for user."}), 400

    crop_details = application["crop_details"]
    farmer_details = application["farmer_details"]

    # Generate Timestamp
    current_datetime = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

    # Blockchain Transaction
    try:
        application_id_hex = application.get("blockchain_id")
        app_id_bytes32 = Web3.to_bytes(hexstr=application_id_hex)

        tx_hash = contract.functions.updateApplicationStatus(
            app_id_bytes32,
            "Application Under Review"
        ).transact({'from': admin_eth_address, 'gas': 6000000})

        tx_receipt = web3.eth.wait_for_transaction_receipt(tx_hash)
        blockchain_id = tx_receipt.transactionHash.hex()

        db['new_reviews'].insert_one({
            "application_id": application_id,
            "blockchain_id": blockchain_id,
            "crop_details": crop_details,
            "farmer_details": farmer_details,
            "eth_address": eth_address,
            "status": "Application Under Review"
        })

        db['stat_applications'].update_one(
            {"_id": ObjectId(application_id)},
            {"$set": {"status": "Application Under Review", "blockchain_id": blockchain_id}}
        )

    except Exception as blockchain_error:
        print(f"Blockchain Error: {str(blockchain_error)}")
        return jsonify({"success": False, "message": "Error updating blockchain status."}), 500
```

Figure 4.3: Send Application For Review

This Flask route is used by the admin to send a crop application for review and notify the farmer. It starts by finding the application in the MongoDB applications collection using its ID. It then looks up the corresponding user to fetch their email.

If the application and user are valid, the smart contract is updated on the blockchain using updateApplicationStatus, changing the status to "Application Under Review". The blockchain transaction hash is saved and stored along with other details in the newreviews and statapplications collections.

Next, the system generates a PDF receipt with key application information like crop name, farmer contact, Ethereum address, and transaction ID. This receipt is attached to an email and sent to the farmer using Flask-Mail.

Finally, the original application is removed from the applications collection (to prevent reprocessing). If email sending fails, the system handles and reports the error.

```python
@app.route('/review_accept', methods=['POST'])
def review_accept():
    db = get_db()

    # Fetch ETH address from the form
    eth_address = request.form.get('eth_address').strip().lower()  # Normalize case

    print(f"Received eth_address: {eth_address}")

    # Fetch application details using ETH address (Case-insensitive)
    application = db['new_reviews'].find_one({
        "eth_address": {"$regex": f"^{eth_address}$", "$options": "i"}
    })

    print(f"Application found: {application}")

    if not application:
        return jsonify({"success": False, "message": "Application not found"}), 404

    # **Check if the application exists on the blockchain**
    try:
        print(f"Checking blockchain for eth_address: {eth_address}")
        exists = contract.functions.applicationExistsByAddress(Web3.to_checksum_address(eth_address)).call()
        print(f"Application exists on blockchain: {exists}")

        if not exists:
            return jsonify({"message": "Application does not exist on the blockchain", "success": False}), 400
    except Exception as e:
        return jsonify({"message": f"Error checking blockchain: {str(e)}", "success": False}), 500

    # **Calculate mid-term check deadline**

    sow_date = datetime.strptime(application['crop_details']['sowDate'], '%Y-%m-%d')
    mid_term_deadline = sow_date + timedelta(days=15)

    # **Update Blockchain Status**
    try:
        tx_hash = contract.functions.updateApplicationStatusByAddress(
            Web3.to_checksum_address(eth_address),
            "Application Under Mid-Term Review"
        ).transact({'from': admin_eth_address, 'gas': 6000000})

        tx_receipt = web3.eth.wait_for_transaction_receipt(tx_hash)
        blockchain_tx_id = tx_receipt.transactionHash.hex()

        # **Insert into midterm_application collection**
        db['midterm_application'].insert_one({
            "application_id": str(application["_id"]),
            "blockchain_id": blockchain_tx_id,
            "crop_details": application["crop_details"],
            "farmer_details": application["farmer_details"],
            "eth_address": application["eth_address"],
            "status": "Application Under Mid-Term Review",
            "mid_term_deadline": mid_term_deadline,
            "mid_term_scheduled": False  # No booking yet
        })

        # **Update stat_applications collection status**
        db['stat_applications'].update_one(
            {"_id": application["_id"]},
            {"$set": {
                "status": "Application Under Mid-Term Review",
                "blockchain_id": blockchain_tx_id,
                "mid_term_deadline": mid_term_deadline,
                "mid_term_scheduled": False
            }}
        )
```

Figure 4.4: Application Accept By Review Team

This Flask route is triggered when the review team accepts a farmer's crop application for the next stage. It starts by retrieving the application based on the Ethereum address provided in the form, using a case-insensitive search.

Next, it verifies whether the application exists on the blockchain by calling a smart contract

function. If it does, the application status is updated on the blockchain to "Application Under Mid-Term Review."

A mid-term deadline is then calculated — 15 days after the sowing date — to ensure timely inspection. The application is stored in the midtermapplication collection with this deadline and marked as not yet scheduled.

The status in the statapplications collection is also updated, and the application is removed from the newreviews collection to avoid duplication. Finally, a notification is sent to the farmer instructing them to book their verification before the deadline.

This function helps move the application to the mid-term verification phase, keeping both the blockchain and database in sync.

```python
@app.route('/grade_crop', methods=['POST'])
def grade_crop():
    """Handles the mid-crop grading process with debugging"""
    try:
        db = get_db()
        data = request.json

        eth_address = data.get("eth_address")  # Use ETH address instead
        grade = data.get("grade")

        if not eth_address or not grade:
            return jsonify({"success": False, "message": "Missing ETH address or grade"}), 400

        # **Retrieve Application from Database**
        application = db["midterm_applications_stage2"].find_one({"eth_address": eth_address})
        if not application:
            return jsonify({"success": False, "message": "Application not found in database"}), 404

        #  *  **FETCH THE APPLICATION ID FROM BLOCKCHAIN**
        blockchain_application_id = contract.functions.ethToApplicationId(Web3.to_checksum_address(eth_address)).call()

        if blockchain_application_id == Web3.to_bytes(hexstr="0x" + "0" * 64):  # Empty bytes32 check
            return jsonify({"success": False, "message": "Application ID not found on blockchain"}), 400

        print(f"Blockchain Application ID: {Web3.to_hex(blockchain_application_id)}")

        # **Check if Application Exists on Blockchain**
        exists_on_chain = contract.functions.applicationExistsByAddress(Web3.to_checksum_address(eth_address)).call()
        if not exists_on_chain:
            return jsonify({"success": False, "message": "Application does not exist on blockchain"}), 400

        # **Check If Already Graded**
        stored_application = contract.functions.cropApplications(blockchain_application_id).call()
        current_grade_status = stored_application[9]  # `graded` field in Solidity struct


        if current_grade_status:
            return jsonify({"success": False, "message": "Application already graded"}), 400

        # **Update on Blockchain**
        tx_hash = contract.functions.gradeCrop(blockchain_application_id, grade).transact({
            'from': web3.eth.accounts[0],
            'gas': 6000000
        })
        web3.eth.wait_for_transaction_receipt(tx_hash)

        # **Update MongoDB**
        # **Update MongoDB (`midterm_applications_stage2`)**
        db["midterm_applications_stage2"].update_one(
            {"eth_address": eth_address},
            {"$set": {
            "status": f"Mid Crop Assessment Done Successfully and Grade Given is: {grade}",
            "grade": grade  # Store grade separately
             }}
            )


# **Update stat_applications Status**
        stat_update_result = db["stat_applications"].update_one(
            {"ETH_ADDRESS": eth_address},
            {"$set": {
                "status": f"Mid Crop Assessment Done Successfully and Grade Given is: {grade}",
                "grade": grade  # Store grade separately
                }}
            )
```

Figure 4.5: Crop Grading

This route (/gradecrop) handles the grading of a farmer's crop after the mid-term field

inspection. It receives the Ethereum address and grade from the frontend, then validates the data. It first checks if the crop application exists in the local MongoDB database and on the blockchain. If valid, it checks whether the crop has already been graded to avoid duplication.

The smart contract method gradeCrop is called to record the grade on the blockchain, ensuring immutability and trust. After successful grading, the application status is updated in the MongoDB collections (midtermapplicationsstage2 and statapplications) with the grade and new status.

To notify the farmer, an email and SMS are sent using the contact info in the database, confirming the completion of the assessment and reminding them to book the harvest within 90 days. The system ensures both transparency and timely communication with the farmer during the grading process.

```python
@app.route('/assign_per_kg_rate', methods=['POST'])
def assign_per_kg_rate():
    """Assigns a per KG rate and quantity to a harvest application."""
    try:
        db = get_db()
        data = request.json

        eth_address = data.get("eth_address")
        rate_per_kg = data.get("rate_per_kg")
        quantity = data.get("quantity")

        if not eth_address or not rate_per_kg or not quantity:
            print("Missing Field: eth_address={}, rate={}, quantity={}".format(eth_address, rate_per_kg, quantity))  # Debugging
            return jsonify({"success": False, "message": "Missing ETH address, rate, or quantity"}), 400
        # **Check if Application Exists on Blockchain**
        exists_on_chain = contract.functions.applicationExistsByAddress(Web3.to_checksum_address(eth_address)).call()
        if not exists_on_chain:
            return jsonify({"success": False, "message": "Application does not exist on blockchain"}), 400

        # **Fetch Application ID from Blockchain**
        application_id_bytes32 = contract.functions.ethToApplicationId(Web3.to_checksum_address(eth_address)).call()
        if application_id_bytes32 == Web3.to_bytes(hexstr="0x" + "0" * 64):  # Empty bytes32 check
            return jsonify({"success": False, "message": "Application ID not found on blockchain"}), 400

        print(f"Blockchain Application ID: {Web3.to_hex(application_id_bytes32)}")

        # **Check if Rate is Already Assigned**
        stored_application = contract.functions.cropApplications(application_id_bytes32).call()
        if stored_application[8]:  # `rateAssigned` field in Solidity struct
            return jsonify({"success": False, "message": "Rate already assigned"}), 400

        # **Call Blockchain Function to Assign Rate & Quantity**
        tx_hash = contract.functions.assignPerKgRate(
            application_id_bytes32, int(rate_per_kg), int(quantity)
        ).transact({'from': web3.eth.accounts[0], 'gas': 6000000})

        web3.eth.wait_for_transaction_receipt(tx_hash)

        # **Fetch Current Grade from Status Before Updating**
        application = db["midterm_applications_stage2"].find_one({"eth_address": eth_address})
        if not application:
            return jsonify({"success": False, "message": "Application not found in database"}), 404

        previous_status = application.get("status", "Mid Crop Assessment Done Successfully and Grade Given is: Unknown")
        grade = previous_status.split("Grade Given is: ")[-1]  # Extract grade from previous status

        farmer_details = application.get("crop_details", {}).get("farmer_details", {})
        farmer_phone = application.get("farmer_details", {}).get("contactNumber")

        # **Update MongoDB (`midterm_applications_stage2`)**
        new_status = f"Application is accepted and Grade given is {grade}, rate assigned per kg is {rate_per_kg} Rs for quantity {quantity} kg."

        db["midterm_applications_stage2"].update_one(
            {"eth_address": eth_address},
            {"$set": {"per_kg_rate": rate_per_kg, "quantity": quantity, "status": new_status}}
        )

        # **Update MongoDB (`stat_applications`)**
        db["stat_applications"].update_one(
            {"ETH_ADDRESS": eth_address},
            {"$set": {"status": new_status}}
        )
        # **Fetch Farmer's Email and Phone Number**
        user_details = db["users"].find_one({"eth_address": eth_address})
        farmer_email = user_details.get("email") if user_details else None
```

Figure 4.6: Assign Rate

This Flask route is used by the admin or review team to assign a price per KG and a quantity for a farmer's crop after the grading process. It begins by verifying the ETH address, rate, and quantity values from the request. Then, it checks if the crop application exists on the

blockchain and whether a rate has already been assigned.

If not, the smart contract function assignPerKgRate() is called to update the rate and quantity on the blockchain. The backend then updates the local MongoDB collections (midtermapplicationsstage2 and statapplications) to reflect this change and forms a status string describing the assigned grade, rate, and quantity.

The system also sends a notification email and SMS to the farmer informing them of the assigned rate and urging them to accept the offer and provide bank details within 5 days. This ensures clear communication and timely processing of applications in the crop supply chain system.

4.2 Steps to access the System

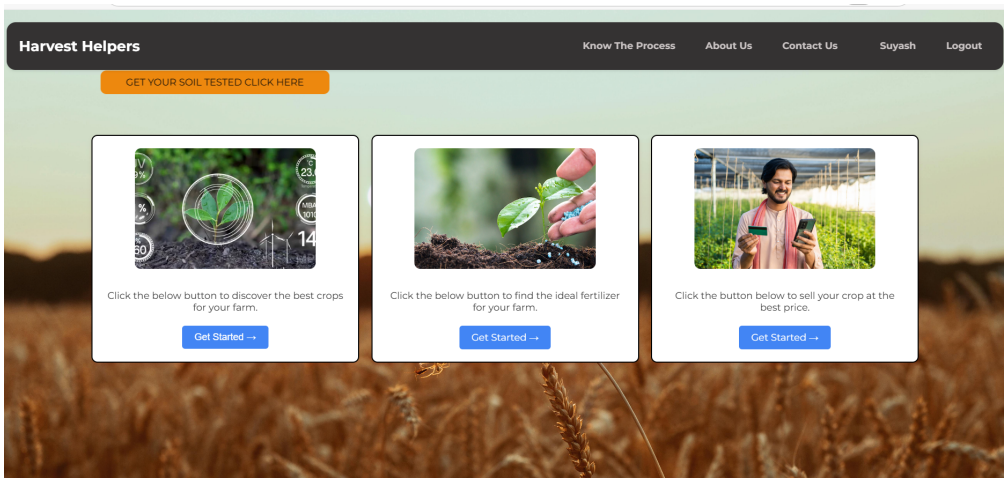


Figure 4.7: Dashboard

Figure 4.2 is Harvest Helpers's dashboard users can log in according to their respective roles, whether they are farmers, members of the review team, or administrators. The page offers three options: "Fertilizer Prediction," "Crop Prediction," and "Apply for Sale of Crop." To access these pages, farmers must be registered.

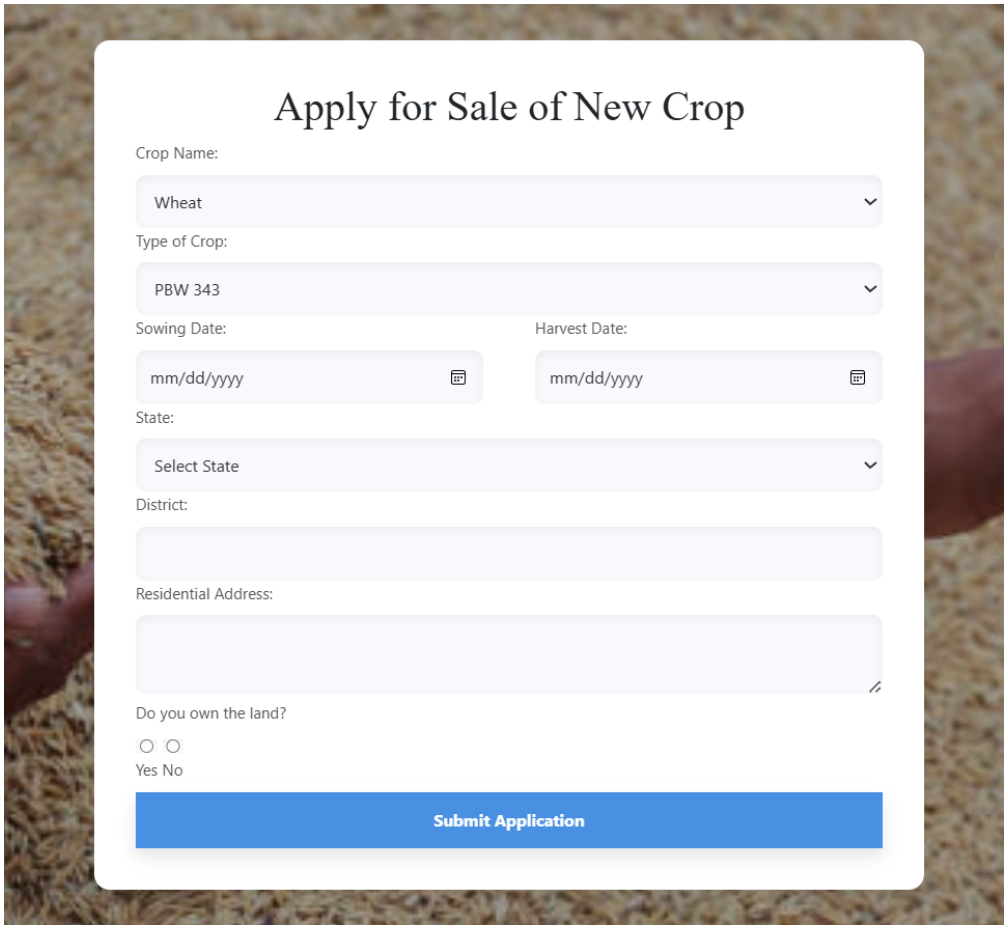


Figure 4.8: Apply For Sale Form

In Figure 4.3, we have the Apply For Sale Form, which requires the farmer to provide details about the crop and personal information to facilitate the sale of the crop at a better marketplace.

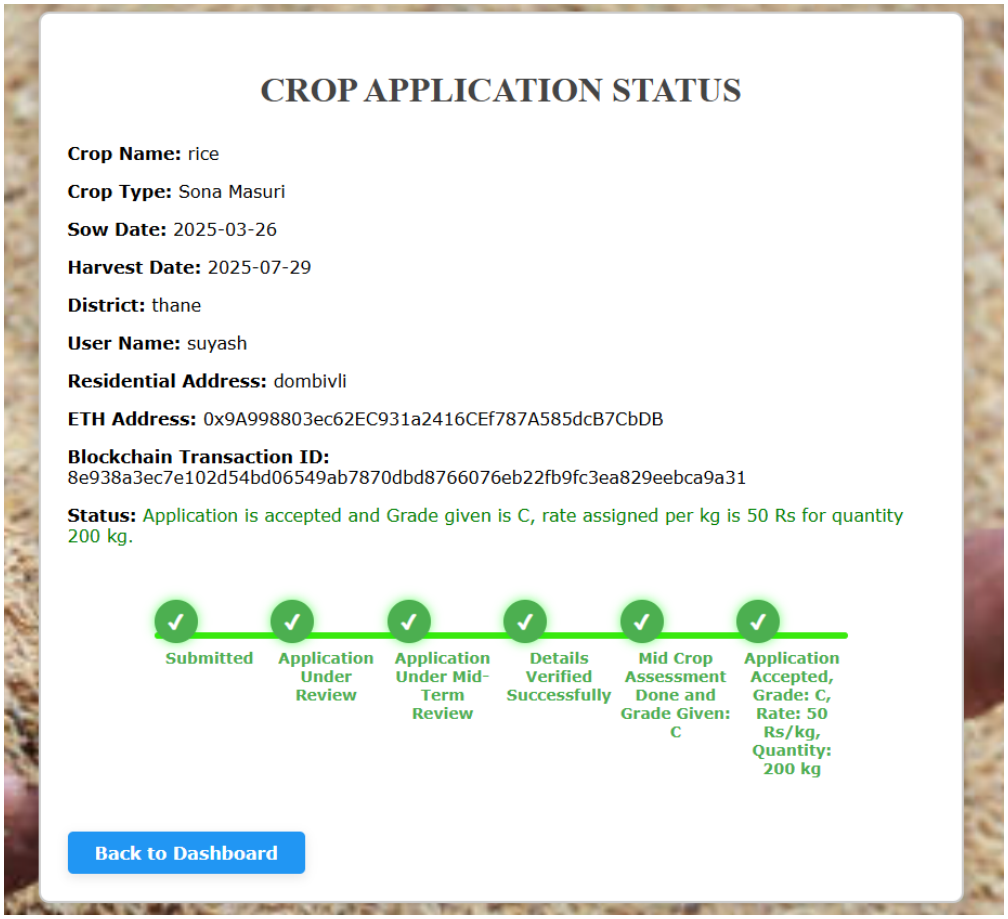


Figure 4.9: Status Tracking Page For Farmers

The Status Page Figure 4.4 in the Harvest Helper project provides real-time tracking of crop applications. It displays key details like crop information, farmer details, blockchain transaction ID, and the current status of the application. A progress bar with circles visually represents each stage, marking completed steps with a tick. This feature helps farmers stay updated on their application's progress, ensuring transparency and smooth communication throughout the approval and assessment process.

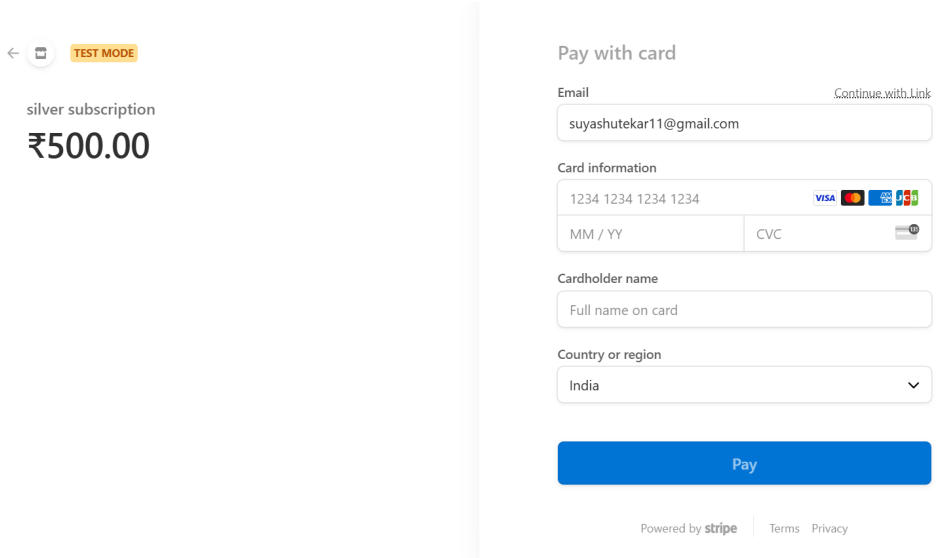


Figure 4.10: Payment Gateway For Subscription

Figure 4.5 is the payment gateway to get subscription.

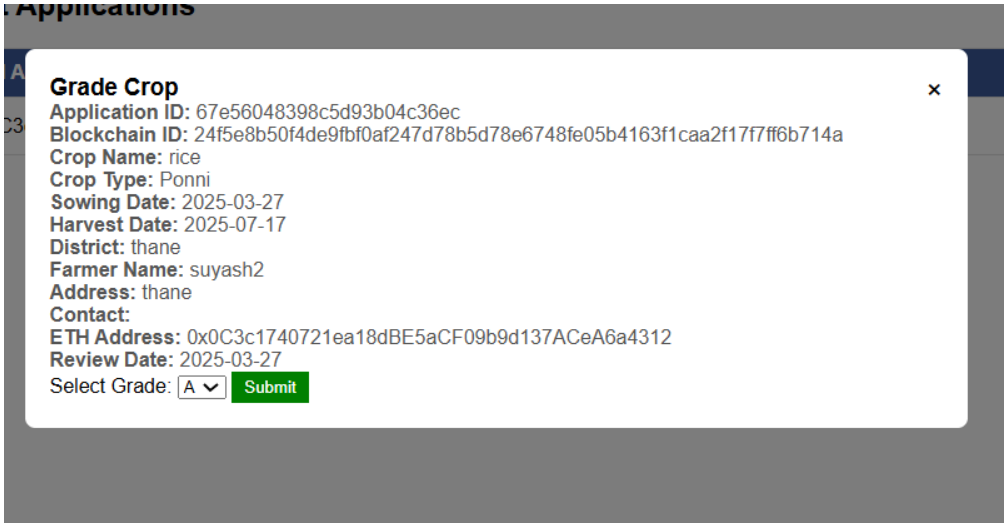


Figure 4.11: Crop Grading

In Figure 4.6 grading is done by the Review Team.Grading is essential in determining price of the crop.

Figure 4.12: Rate and Quantity Assignment

In Figure 4.7 rate is assigned to the crop by the review team by checking the quality of crop after harvest and also the grade given during mid term assessment.Also the quantity of crop available for sale is checked and entered in to the system.

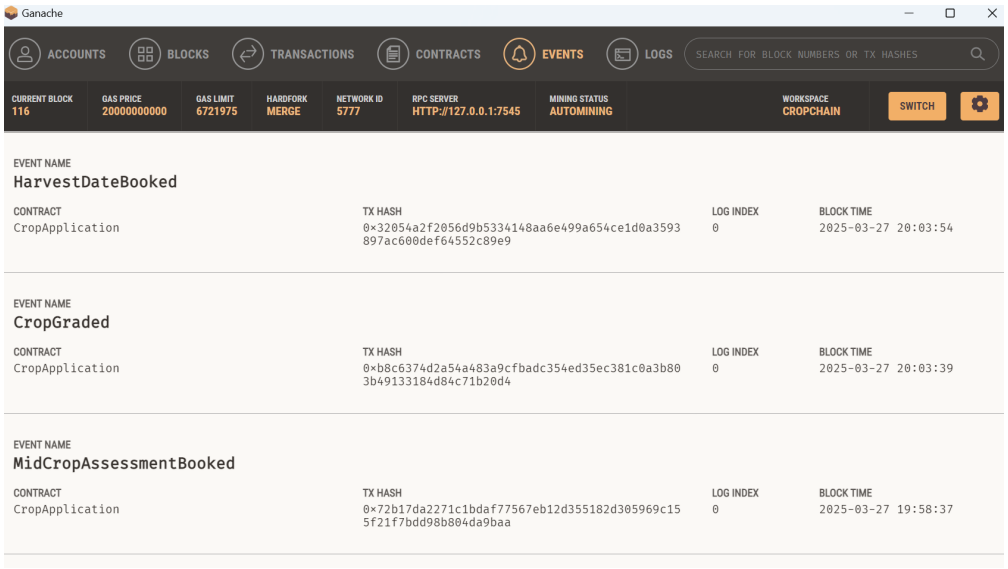


Figure 4.13: Recorded Data On Ganache

Figure 4.8 shows the events happened on blockchain through ganache software.

4.3 Timeline Sem VIII

GANTT CHART BE Major Project 1B

Smartsheet Tip → A Gantt chart's visual timeline allows you to see details about each task as well as project

PROJECT TITLE	Harvest Helper: Smart Agriculture Powered by Blockchain and Machine Learning
PROJECT GUIDE	Ms. Shraddha Birje

DEPARTMENT NAME : Department of Information Technology

DATE 19/10/2024

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION (Weeks)	PCT OF TASK COMPLETE
1	Project Conception and Initiation					
1.1	Area of Problem Search	Suyash	22/7/24	27/7/24	1	100%
1.2	Project Title	Suyash	29/7/24	2/8/2024	1	100%
1.3	Abstract	Suyash,Atharva	29/7/24	2/8/2024		100%
1.4	Objectives	Suyash,Atharva,Yash	5/8/2024	9/8/2024		100%
1.5	Literature Review	Suyash,Atharva,Shrikant	12/8/2024	26/8/24	3	100%
1.6	Problem Definition	Suyash,Atharva	29/8/24	2/9/2024	1	100%
1.7	Scope	Shrikant,Yash	26/8/24	2/9/2024	1	100%
1.8	Technology stack	Suyash	26/8/24	2/9/2024	1	100%
1.9	Benefits for environment	Suyash	5/9/2024	9/9/2024	1	100%
1.1	Benefits for Society	Suyash,Atharva,Yash	5/9/2024	9/9/2024	1	100%
1.11	Applications	Suyash,Atharva	12/9/2024	16/9/24	1	100%
2	Project Design					
2.1	Proposed System	Suyash,Atharva	12/09/2024	16/9/24	1	100%
2.2	Flow Diagram	Suyash,Atharva	19/09/24	23/9/24	1	100%
2.3	System Architecture	Suyash,Atharva	19/9/24	23/9/24	2	100%
2.4	Design(Flow Of Modules)	Suyash	19/9/24	23/9/24	2	100%
3	Synopsis Preparation					
3.1	Documentation	Suyash,Atharva,Shrikant	24/06/2024	10/07/2024	1	100%

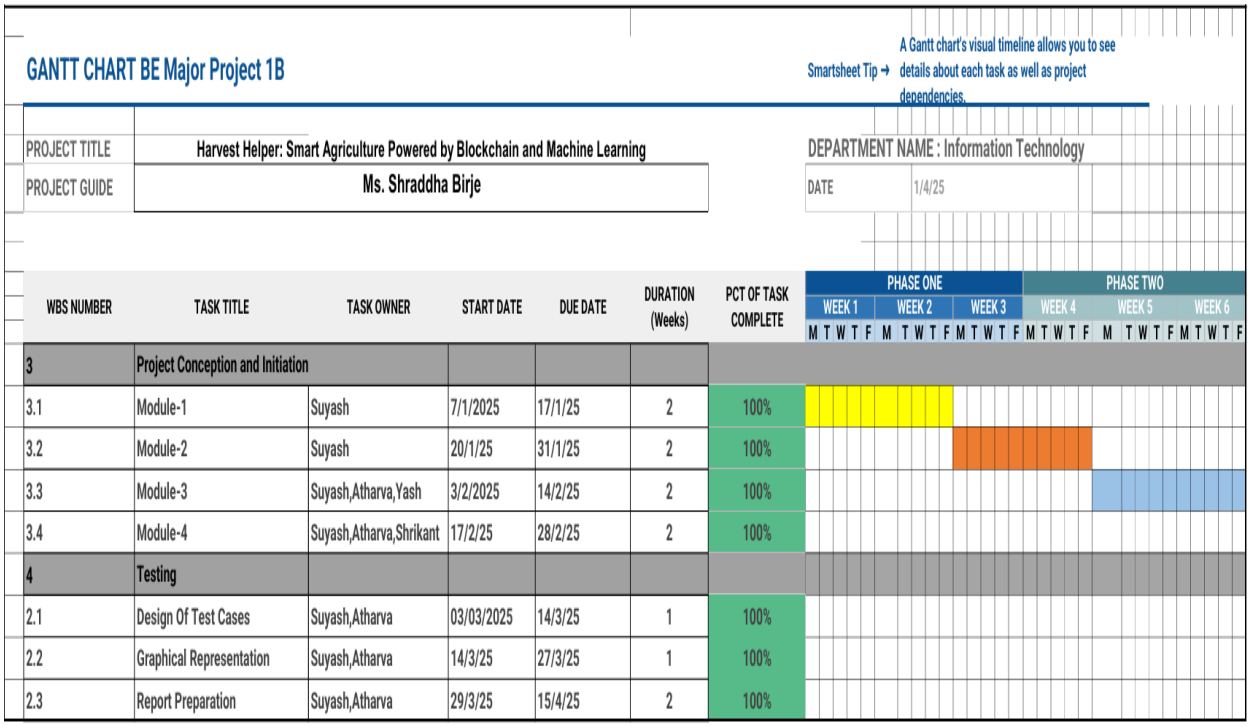
PHASE ONE: WEEK 1 (M T W T F) WEEK 2 (M T W T F) WEEK 3 (M T W T F)
PHASE TWO: WEEK 4 (M T W T F) WEEK 5 (M T W T F) WEEK 6 (M T W T F)

Figure 4.14: Timeline Sem VII Part 1

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION (Weeks)	PCT OF TASK COMPLETE
1	Project Conception and Initiation					
1.1	Area of Problem Search	Suyash	22/7/24	27/7/24	1	100%
1.2	Project Title	Suyash	29/7/24	2/8/2024	1	100%
1.3	Abstract	Suyash,Atharva	29/7/24	2/8/2024		100%
1.4	Objectives	Suyash,Atharva,Yash	5/8/2024	9/8/2024		100%
1.5	Literature Review	Suyash,Atharva,Shrikant	12/8/2024	26/8/24	3	100%
1.6	Problem Definition	Suyash,Atharva	29/8/24	2/9/2024	1	100%
1.7	Scope	Shrikant,Yash	26/8/24	2/9/2024	1	100%
1.8	Technology stack	Suyash	26/8/24	2/9/2024	1	100%
1.9	Benefits for environment	Suyash	5/9/2024	9/9/2024	1	100%
1.1	Benefits for Society	Suyash,Atharva,Yash	5/9/2024	9/9/2024	1	100%
1.11	Applications	Suyash,Atharva	12/9/2024	16/9/24	1	100%
2	Project Design					
2.1	Proposed System	Suyash,Atharva	12/09/2024	16/9/24	1	100%
2.2	Flow Diagram	Suyash,Atharva	19/09/24	23/9/24	1	100%
2.3	System Architecture	Suyash,Atharva	19/9/24	23/9/24	2	100%
2.4	Design(Flow Of Modules)	Suyash	19/9/24	23/9/24	2	100%
3	Synopsis Preparation					
3.1	Documentation	Suyash,Atharva,Shrikant	24/06/2024	10/07/2024	1	100%

PHASE THREE: WEEK 7 (M T W T F) WEEK 8 (M T W T F) WEEK 9 (M T W T F)
PHASE FOUR: WEEK 10 (M T W T F) WEEK 11 (M T W T F) WEEK 12 (M T W T F)

Figure 4.15: Timeline Sem VII Part 2



GANTT CHART BE Major Project 1B

Smartsheet Tip → A Gantt chart's visual timeline allows you to see details about each task as well as project dependencies.

PROJECT TITLE: Harvest Helper: Smart Agriculture Powered by Blockchain and Machine Learning

PROJECT GUIDE: Ms. Shraddha Birje

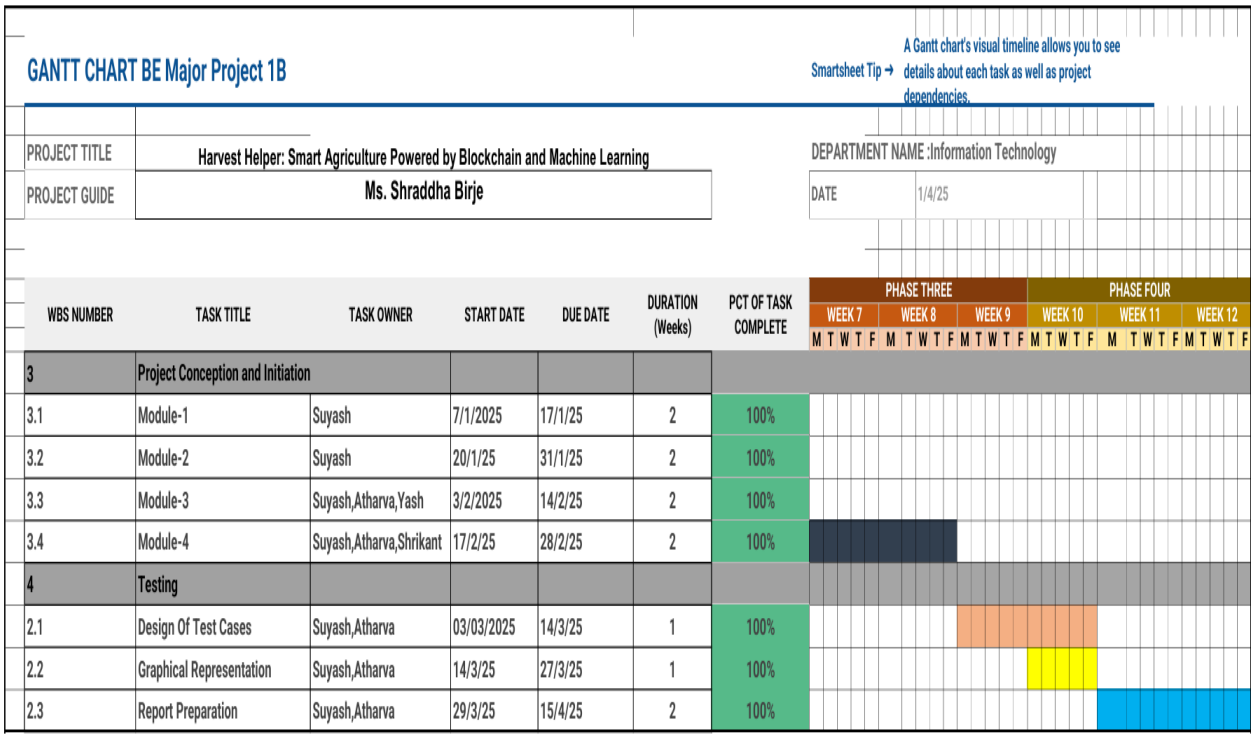
DEPARTMENT NAME : Information Technology

DATE: 1/4/25

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION (Weeks)	PCT OF TASK COMPLETE
3	Project Conception and Initiation					
3.1	Module-1	Suyash	7/1/2025	17/1/25	2	100%
3.2	Module-2	Suyash	20/1/25	31/1/25	2	100%
3.3	Module-3	Suyash,Atharva,Yash	3/2/2025	14/2/25	2	100%
3.4	Module-4	Suyash,Atharva,Shrikant	17/2/25	28/2/25	2	100%
4	Testing					
2.1	Design Of Test Cases	Suyash,Atharva	03/03/2025	14/3/25	1	100%
2.2	Graphical Representation	Suyash,Atharva	14/3/25	27/3/25	1	100%
2.3	Report Preparation	Suyash,Atharva	29/3/25	15/4/25	2	100%

PHASE ONE: WEEK 1, WEEK 2, WEEK 3 — PHASE TWO: WEEK 4, WEEK 5, WEEK 6

Figure 4.16: Timeline Sem VIII Part 1



GANTT CHART BE Major Project 1B

Smartsheet Tip → A Gantt chart's visual timeline allows you to see details about each task as well as project dependencies.

PROJECT TITLE: Harvest Helper: Smart Agriculture Powered by Blockchain and Machine Learning

PROJECT GUIDE: Ms. Shraddha Birje

DEPARTMENT NAME :Information Technology

DATE: 1/4/25

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION (Weeks)	PCT OF TASK COMPLETE
3	Project Conception and Initiation					
3.1	Module-1	Suyash	7/1/2025	17/1/25	2	100%
3.2	Module-2	Suyash	20/1/25	31/1/25	2	100%
3.3	Module-3	Suyash,Atharva,Yash	3/2/2025	14/2/25	2	100%
3.4	Module-4	Suyash,Atharva,Shrikant	17/2/25	28/2/25	2	100%
4	Testing					
2.1	Design Of Test Cases	Suyash,Atharva	03/03/2025	14/3/25	1	100%
2.2	Graphical Representation	Suyash,Atharva	14/3/25	27/3/25	1	100%
2.3	Report Preparation	Suyash,Atharva	29/3/25	15/4/25	2	100%

PHASE THREE: WEEK 7, WEEK 8, WEEK 9 — PHASE FOUR: WEEK 10, WEEK 11, WEEK 12

Figure 4.17: Timeline Sem VIII Part 2

Chapter 5

Testing

5.1 Software Testing

Software testing is a crucial phase in the development of the **Harvest Helper** project. It ensures that the system functions correctly, meets the specified requirements, and is free from defects. Various testing methodologies exist, but for this project, a combination of **Functional Testing, Integration Testing, and Security Testing** has been chosen to ensure the accuracy and security of blockchain transactions, data consistency, and user interactions.

Functional testing is the most suitable approach for this project because it verifies that all features work according to their intended specifications. Since the project involves multiple stakeholders (farmers, reviewers, and administrators), it is essential to ensure that each user role performs its designated tasks correctly. Furthermore, due to blockchain integration, security testing is vital to prevent unauthorized access and ensure transaction integrity.

5.2 Functional Testing

Functional Testing is used to validate that the **Harvest Helper** system meets all the functional requirements specified during the design phase. This method involves testing different modules of the application to ensure that each function works correctly according to predefined inputs and expected outputs.

For this project, functional testing was selected because it ensures that key processes, such as *crop application submission, mid-term check booking, crop grading, and price assignment*, function without errors. Since blockchain transactions play a crucial role in recording and verifying data, functional testing ensures that all data flows, UI interactions, and backend processes work smoothly. The results of functional testing confirm that the system operates as intended, providing a reliable platform for farmers and reviewers.

Test Cases

The following table outlines key functional test cases performed during the testing phase:

Test Case ID	Test Description	Expected Output	Actual Output	Status
TC-001	User Registration	Account should be created	Successfully created	**Pass**
TC-002	User Login	Valid users should log in, invalid users should be rejected	Correct behavior observed	**Pass**
TC-003	Crop Application Submission	Application should be stored in MongoDB and blockchain	Successfully recorded	**Pass**
TC-004	Mid-Term Review Booking	Farmers should be able to book a mid-term check appointment	Successfully booked	**Pass**
TC-005	Smart Contract Transactions	Status updates and grading should be stored on blockchain	Transactions recorded correctly	**Pass**
TC-006	Email Notifications	Users should receive email notifications for updates	Emails sent successfully	**Pass**
TC-007	Harvest Date Booking	Farmers should select a harvest date	Harvest date recorded	**Pass**
TC-008	Crop Grading	Reviewers should assign grades	Grade stored correctly	**Pass**
TC-009	Unauthorized Access Prevention	Non-admin users should not access admin features	Unauthorized users restricted	**Pass**

Table 5.1: Functional Testing Results for Harvest Helper

Chapter 6

Result and Discussions

The **Harvest Helper** project successfully integrates **blockchain technology** with a **Flask-based web application** to streamline the **crop application, mid-term verification, and harvest process**. This chapter evaluates the investigation carried out, presents key findings, and discusses the contributions of the study. The discussion logically leads to inferences, conclusions, and potential areas for future enhancements.

6.0.1 Evaluation of the Investigation

The investigation aimed to develop an **automated, transparent, and immutable crop verification system** using **Ethereum-based smart contracts**. The study focused on:

Blockchain-Powered Crop Application Processing

- Farmers submit applications through the Flask web interface, which are **stored both in MongoDB and on the Ethereum blockchain**.
- Each application receives a **unique blockchain transaction ID**, ensuring traceability and preventing tampering.
- Smart contracts enable **status tracking**, with updates recorded immutably on-chain.

Mid-Term Crop Assessment Automation

- Once an application reaches the **mid-term review stage**, farmers must **book an assessment appointment**, ensuring timely crop verification.
- The **Flask backend listens for blockchain events** to automatically update application statuses.
- The review team evaluates crop conditions and **assigns a grade (A, B, C)** via the web interface, with the data permanently stored on the blockchain.

Smart Contract-Driven Harvest and Pricing Mechanism

- Farmers **schedule harvest dates** after mid-term verification.
- On the scheduled date, the **application automatically appears in the review team's dashboard**.
- The review team assigns a **per KG rate** to the crop, which is **permanently recorded on the blockchain**, ensuring pricing transparency.

Flask and Web3.py Integration for Real-Time Updates

- The Flask backend is integrated with **Web3.py** to track smart contract events and trigger automatic status updates.
- This ensures that applications **seamlessly transition between different process stages** based on scheduled dates.

6.0.2 Key Contributions of the Study

This project highlights how blockchain technology can enhance **trust, efficiency, and transparency** in the agricultural sector. The key contributions include:

- **Tamper-Proof Crop Verification** – All application statuses, mid-term assessments, and pricing decisions are stored immutably on the blockchain.
- **Automation of Workflow** – Farmers and reviewers no longer need manual follow-ups; smart contracts automatically trigger application transitions.
- **Transparent Pricing** – The per KG rate assigned is visible on-chain, ensuring **fair trade practices**.

Chapter 7

Conclusion

The **Harvest Helper** project was designed and developed to streamline and enhance the crop application, assessment, and marketplace process using blockchain technology. The primary objective was to create a transparent, efficient, and tamper-proof system for farmers, reviewers, and administrators to manage crop applications, conduct mid-term assessments, and facilitate crop sales.

Throughout the project, various **software development methodologies, technologies, and testing strategies** were implemented to ensure a robust and functional system. The use of **Flask (Python), MongoDB, HTML, CSS, JavaScript, and Solidity** enabled the development of a secure and scalable web application integrated with blockchain-based smart contracts. The platform ensures that each step, from crop submission to grading and pricing, is recorded on the blockchain, providing an immutable and verifiable transaction history.

Functional testing was conducted extensively to validate system performance, ensuring that key operations such as **user registration, crop submission, mid-term check scheduling, grading, and crop marketplace transactions** work flawlessly. This helped in identifying and rectifying potential issues, improving user experience, and ensuring system reliability. The integration of **smart contracts** not only eliminated manual errors but also increased trust among stakeholders by making data transparent and tamper-proof.

7.1 Key Achievements

The **Harvest Helper** project successfully achieves the following key outcomes:

- **Automated Crop Application Process** – Farmers can submit applications, track progress, and book mid-term reviews with ease.
- **Blockchain-Powered Transparency** – All key transactions, including grading and per KG rate assignment, are securely stored on the blockchain.

- **Efficient Review Mechanism** – The review team can assess crops, assign grades, and approve harvest dates, ensuring an efficient workflow.
- **Seamless E-Commerce Integration** – Crops can be listed for sale, allowing buyers to view transaction histories and make informed purchases.
- **Security & Data Integrity** – Blockchain ensures that no data can be manipulated, providing authenticity and preventing fraud.

Chapter 8

Future Scope

Harvest Helper has huge potential for growth, especially by leveraging blockchain, smart contracts, and AI-driven insights. Here are some future enhancements that can add more value to your system:

- **Full-Fledged Blockchain-Based Marketplace**
  - Smart Contract-Based Auctions: Farmers can auction their crops to get the best price. Buyers place bids, and the highest bidder wins automatically via smart contract.
  - Escrow System for Secure Transactions: Implement a blockchain-based escrow service that holds funds until both parties confirm delivery, ensuring elimination of fraud and increasing buyer confidence.
- **AI-Driven Insights for Farmers**
  - AI-Powered Crop Price Prediction: Use historical data and market trends to predict crop prices and suggest the best time for farmers to sell their crops.
  - Smart Farming Advisory: Integrate AI-based weather prediction to alert farmers about climate risks.
  - Yield Estimation Models: Use satellite and IoT sensor data to predict crop yield.
- **IoT Integration for Smart Farming**
  - Real-Time Crop Monitoring: Connect IoT soil sensors and weather APIs to monitor soil moisture, temperature, and crop health.
  - Automated Irrigation Systems: Use AI + IoT to automate water supply based on real-time weather data.
  - Blockchain-Based Supply Chain: Track crops from sowing to harvesting using RFID and IoT sensors.
- **Multi-Payment System**
  - Multi-Payment Gateway: Integrate UPI, PayPal, or Stripe for transactions.

    - Wallet System for Users: Implement an in-app wallet for buyers and sellers to store funds.

- **Government & Institutional Collaboration**
    - Government Subsidy Integration: Link farmers' profiles to government subsidy programs.
    - Loans & Credit Facilities: Partner with banks or microfinance institutions to provide small loans based on crop yield history.
    - Export Market Integration: Allow verified farmers to sell crops internationally through government export programs.

- **Community & Social Impact**
    - Farmer Social Network: Build a community platform where farmers can share best practices.
    - Agri-Learning & Training Portal: Offer training materials, video tutorials, and live webinars.
    - Crowdfunding for Farmers: Allow people to invest in farms and share profits based on harvest yield.

Bibliography

[1] E. N. Chuka, B. A. Alhassan, and P. C. Ndama, "Transforming Agricultural Supply Chains: Leveraging Blockchain-Enabled Java Smart Contracts and IoT Integration," International Journal of Computer Applications, 2023. DOI: 10.5120/ijca2021612345.

[2] P. Burgess, F. Sunmola, and S. Wertheim-Heck, "Blockchain Enabled Quality Management in Short Food Supply Chains," Procedia Computer Science, vol. 200, pp. 904–913, 2022. DOI: 10.1016/j.procs.2022.01.288.

[3] Y. Zhang, P. White, D. Schmidt, and L. Louw, "Blockchain-Based Agricultural Supply Chain: A Review," Journal of Agriculture and Food Information, vol. 21, no. 2, pp. 123-145, 2020. DOI: 10.1080/10496505.2020.1725894.

[4] R. Holambe, P. Patil, P. Pawar, S. Salunkhe, and H. Joshi, "IOT based Crop Recommendation, Crop Disease Prediction and Its Solution," International Research Journal of Engineering and Technology (IRJET), vol. 07, issue 10, pp. 24-26, 2020. DOI: N/A

[5] C. Singh, R. Thakkar, and J. Warraich, "Blockchain in Supply Chain Management," *European Journal of Engineering and Technology Research*, vol. 7, issue 5, pp. 60-69, 2022.

[6] M. A. Hadi and M. R. Ali, "Study of Blockchain Technology in Farmers Portal," *Mathematical Statistician and Engineering Applications*, vol. 72, no. 1, pp. 1389-1397, 2023.

[7] T. Fidowaty and R. Y. Supriadi, "E-commerce in Farmers Communities," *IOP Conference Series: Materials Science and Engineering*, vol. 879, 012131, 2020. DOI: 10.1088/1757-899X/879/1/012131.

Appendices

8.1 Appendix A

5.1.1 Project Structure

Here is the project directory structure for the Harvest Helpers application:

```
project-root/
|
+-- app/                # Flask-based backend
|   +-- routes/         # Application routes
|   +-- controllers/    # Logic for handling requests
|   +-- models/         # MongoDB models
|   +-- templates/      # HTML templates
|   +-- static/         # CSS, JS, and images
|   +-- config.py       # Configuration settings (DB, etc.)
|   `-- app.py          # Main Flask application file
|
+-- blockchain/         # Blockchain contract scripts and logic
|   `-- contracts/      # Smart contracts for blockchain interactions
|
+-- venv/               # Virtual environment for Flask
+-- requirements.txt    # Python package dependencies
+-- README.md           # Project documentation
+-- tests/              # Unit tests for backend and blockchain
```

5.1.2 Setting Up the Flask Backend

Create a Virtual Environment for Flask

1. Open your terminal and navigate to the project root directory.
2. Run the command to create a virtual environment:

```
python -m venv venv
```

3. Activate the virtual environment:

```
source venv/bin/activate  # On macOS/Linux
.\venv\Scripts\activate   # On Windows
```

Install Flask and Dependencies

1. Create a `requirements.txt` file and add the following dependencies:

```
Flask
Flask-Mail
Flask-Cors
pymongo
```

2. Install the dependencies:

```
pip install -r requirements.txt
```

Set Up the Flask Application

1. In the `app.py` file, set up the Flask server:

```
from flask import Flask
from flask_cors import CORS

app = Flask(__name__)
CORS(app)

@app.route('/')
def home():
    return "Hello from Harvest Helpers!"

if __name__ == '__main__':
    app.run(debug=True)
```

MongoDB Database Setup

1. Install MongoDB locally or set up MongoDB Atlas.
2. Install the MongoDB driver for Python:

```
pip install pymongo
```

3. In the `config.py` file, configure MongoDB.

```
MONGO_URI = "your_mongodb_connection_string"
```

4. Use MongoDB models in your app to store and manage data such as farmer details, crop applications, review reports, and eCommerce transactions

5.1.2 Blockchain Integration (Smart Contracts)

Set Up Smart Contracts

1. Define smart contracts in Solidity or an appropriate blockchain language.
2. Implement contracts for transactions such as crop sale agreements, quality verification, gardes and quantity.

Deploy Smart Contracts

1. Use a blockchain environment like Ethereum or Hyperledger to deploy contracts.
2. Test the blockchain interactions using tools like Truffle or Remix.

5.1.3 Running the Project

To run the Flask backend, activate the virtual environment and use:

```
flask run
```

To interact with the blockchain, run the blockchain node or service and deploy smart contracts.

5.1.4 Additional Technologies

- **MongoDB:** Database used for storing and retrieving user information, crop details, and transaction logs.
- **Flask-Mail:** For sending notifications to users (e.g., farmers, admins, dealers).
- **Blockchain:** Immutable records of crop sales, quality checks, and agreements.

Appendix B: Useful Resources

Here are some useful resources related to the technologies used in the Harvest Helpers project:

- Flask Documentation: https://flask.palletsprojects.com/en/2.0.x
- MongoDB Documentation:https://docs.mongodb.com/manual/
- Python Flask-Mail Guide: https://pythonhosted.org/Flask-Mail/
- Blockchain Resources:https://ethereum.org/en/developers/docs/
- Solidity Documentation:https://docs.soliditylang.org/en/v0.8.0/